

Universiteit
Leiden
The Netherlands

Guiding evolutionary search towards innovative solutions

Reehuis, E.J.J.

Citation
Reehuis, E. J. J. (2013, December 17). *Guiding evolutionary search towards innovative solutions*. Retrieved from https://hdl.handle.net/1887/22875

Version:	Corrected Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/22875

Note: To cite this publication please use the final published version (if applicable).

Cover Page



Universiteit Leiden



The handle http://hdl.handle.net/1887/22875 holds various files of this Leiden University dissertation

Author: Reehuis, Edgar
Title: Guiding evolutionary search towards innovative solutions
Issue Date: 2013-12-17

Guiding Evolutionary Search towards Innovative Solutions

EDGAR J. J. REEHUIS

Guiding Evolutionary Search towards Innovative Solutions

Proefschrift

ter verkrijging van

de graad van Doctor aan de Universiteit Leiden,

op gezag van Rector Magnificus prof. mr. C.J.J.M. Stolker,

volgens besluit van het College voor Promoties

te verdedigen op dinsdag 17 december 2013

klokke 8.45 uur

door

Edgar Julius Jeronimus Reehuis

geboren te Breda, in 1982

Promotiecommissie

Promotoren	Prof. dr. T.H.W. Bäck	
	Prof. dr. B.A. Sendhoff	*Technische Universität Darmstadt*
Overige leden	Prof. dr. F.M.E. Duddeck	*Technische Universität München*
	Prof. dr. J.N. Kok	
	Prof. dr. B.R. Katzy	
	Dr. M.T.M. Emmerich	
	Dr. W.A. Kosters	

Het onderzoek beschreven in dit proefschrift is uitgevoerd aan het Leiden Institute of Advanced Computer Science (LIACS, Universiteit Leiden) en gedeeltelijk aan het Honda Research Institute Europe (HRI-EU).

Contents

1 Introduction **1**

1.1 Research Goals and Contribution of this Thesis 2

1.2 Thesis Outline 3

2 Preliminaries **5**

2.1 Evolutionary Algorithms 6

2.1.1 Covariance Matrix Adaptation Evolution Strategy (CMA-ES) . . . 7

2.2 Approximation Models 11

2.3 Surrogate-assisted Optimization 14

2.3.1 Surrogate Utility Functions 15

2.4 Summary 18

3 Innovativeness in Design Optimization **19**

3.1 Novelty, Innovativeness, and Interestingness 21

3.2 Search Objective: Towards Innovative Solutions through Diversity 27

3.2.1 Measuring Performance 28

3.3 Efficiently Finding Diverse High-quality Solutions 31

3.4 Summary 33

4 Criteria for Guiding Exploration **35**

4.1 Exploration Measures in Adaptive Search 38

4.2 Novelty 40

4.2.1 Distance-based Novelty 41

4.2.2 Learning-based Novelty 46

4.3 Interestingness 55

4.3.1 Notational Scheme 56

4.3.2 Regional Error Value 57

4.3.3 Interestingness Measures 60

4.4 Summary 71

5 Integration of Exploration Criteria in the Search **75**
5.1 Integration Schemes for Scenarios with Infeasibility 77
5.2 Multiobjective Selection 80
5.2.1 Weighted Aggregation 80
5.2.2 Pareto-based Selection 82
5.3 Separate Exploration Phases 85
5.3.1 Alternating Restart Scheme 86
5.4 Summary 95

6 Application to Airfoil Optimization **97**
6.1 Domain-specific Distance Measure 98
6.2 Test Case: Two-dimensional Stator-blade Optimization 100
6.2.1 Quality Function 101
6.3 Modus Operandi 103
6.4 Results 105
6.5 Summary 112

7 Conclusions and Outlook **115**
7.1 Summary 115
7.2 Outlook 119

Bibliography **121**

Samenvatting **129**

Acknowledgments **133**

About the Author **135**

1

Introduction

In *optimization*, the goal is to find an input for which a given mathematical function produces the *optimal* output value. The optimal output value is either the minimal or maximal value that the function can return, a choice that is specified in advance. A real-world engineering design task, such as shaping an airfoil, can be viewed as a *quality-optimization problem*: The input is the proposed design and the output its assessed quality. A design is then referred to as a *solution* for the optimization problem.

For optimizing a design task in an automated way, the input has to be re-formulated in such a way, i.e., *encoded*, that it is possible to automatically vary and adjust proposed solutions. A solution is, for instance, represented by a sequence of numbers, from which the actual design is retrieved using a *decoding* scheme. Furthermore, the quality of a design has to be quantified into a numerical score as well. Ideally, an automated pipeline is set up in which the numerical input is fed to a *simulator* that models the represented design's real-world performance in a simulated environment, and approximates its actual quality score. The simulator, accepting solutions as input and producing a quality score as output, effectively takes on the role of a *quality function*.

Given this numerical representation of solutions and the available automated quality assessment, an *optimization algorithm* can be applied that, without human interaction, aims to find an optimal solution to the design task by systematically varying the numerical input. Note that this optimality is restricted by the freedom in varying the designs that is allowed through the used encoding, and relates strictly to the simulated environment and the used quality quantification.

Evolutionary Algorithms (EAs) are methods that approach optimizing a function by using a *population* of, e.g., numerically-represented, solutions that *evolve* towards an optimal configuration. Unlike classical, *deterministic* optimization algorithms, the

stochastic EAs rely on randomness to drive the optimization process. The to-be-optimized function provides feedback on *solution fitness*, which is used to select the most-fit solutions and thereby steer the random sampling. While the population-based EAs are relatively slow with respect to the required number of function evaluations in approaching an optimum, their moderate robustness to deceptive characterisics of the function landscape and flexibility in applying them using varying types of problem formulations make that they are frequently considered for real-world optimization tasks.

Optimization methods in general, and EAs specifically, are able to assist in uncovering *innovative solutions*, roughly defined as solutions involving unconventional new ways in obtaining good performance. By automatically varying solutions, a different path may be embarked on towards high quality than would be considered by a human engineer guided by his or her education. Nevertheless, even in using a population of solutions, an optimization method most-likely converges to only one optimized solution at a time. Due to their stochastic nature, however, different runs of an EA *are* potentially able to produce alternative solutions.

Exploration-inducing schemes exist that operate on top of optimization algorithms and make them actively target multiple high-performing solutions. We aim for exploration to be performed in a sequential, path-based fashion, allowing for application to scenarios in which optimization needs to be started from a single, fixed initial solution. An additional exploration criterion is to be integrated into the evolutionary search, next to quality, such that multiple high-quality solutions can be found in a single run, assisted by a memory of solutions encountered earlier during the search. Not playing a role in the active exploration, quality optimization forms an important, but supportive part of the overall process, which is therefore termed “search” instead of “optimization”.

1.1 · Research Goals and Contribution of this Thesis

The main goal of this work is to present a method, running on top of an underlying optimization algorithm, that increases its likelihood of uncovering innovative solutions. As such, a way of measuring the capability of a method to do so will be presented, based on a definition of innovativeness of solutions in design optimization tasks.

Different exploration criteria are described in literature that are based on a knowledge base of solutions encountered earlier in the search. In this work, these are aimed

to be used for inducing sequential exploration in design spaces, adjusting them where necessary. Furthermore, the different exploration criteria are brought together in a uniform naming and notational scheme, to facilitate implementation and comparison with this aim in mind.

After formulating the exploration criteria, an appropriate way of how to integrate such a criterion in the search, next to quality, is to be presented. Lastly, the developed methods are tested on a real-world design optimization task, showing the steps required in doing so, and for verification of their ability of assisting in finding innovative solutions.

1.2 · Thesis Outline

Based on the goals described above, this thesis takes the reader stepwise from a definition of innovativeness of solutions to application of a scheme for assisting in finding innovative solutions to a real-world airfoil optimization task:

- Chapter 2 introduces underlying techniques that will be applied in this work or that will be used as inspiration, discussing Evolutionary Algorithms, *approximation models*, and *surrogate-assisted search*;

- Chapter 3 discusses innovativeness in design optimization, relating it to *novelty* and *interestingness* of solutions. Furthermore, a scheme for measuring performance of a method aimed at assisting in finding innovative solutions is laid out, published on before in [**Reehuis et al., 2013b**] and [**Reehuis et al., 2013c**];

- Chapter 4 formulates criteria for inducing exploration in design spaces, reusing the ideas of novelty and interestingness. An initial comparison of the different criteria is performed on an artificial setup, to select the most-promising measures that will be considered for integration in quality-based search. Parts of this chapter were published before in [**Reehuis et al., 2011**], [**Reehuis et al., 2013a**], [**Reehuis et al., 2013b**], and [**Reehuis et al., 2013c**];

- Chapter 5 discusses different ways of integrating an exploration criterion in quality-based search. An initial comparison is performed on an artificial setup involving, like the real-world airfoil optimization task, the requirement of starting from a single fixed solution, and has large areas with *infeasible*, i.e., intolerable, solutions. This

chapter continues on earlier work in [**Reehuis et al., 2011**], [**Reehuis et al., 2013b**], and [**Reehuis et al., 2013c**];

- Chapter 6: Two variants of a method for assisting in finding innovative solutions are applied to a real-world optimization task, described separately in [**Reehuis et al., 2013b**] and in [**Reehuis et al., 2013c**], and their performance is statistically compared;

- Chapter 7 summarizes the thesis and provides future research directions.

2

Preliminaries

The overall objective of this thesis is to derive methods that increase the likelihood of discovery of *innovative solutions*, in using automated search. This chapter provides an overview of established techniques and approaches that will be applied or from which ideas are derived, reported on in later chapters. As underlying search method, an advanced *Evolutionary Algorithm* will be used that adapts its search distribution to the search landscape, the *CMA-ES*. It is introduced in Section 2.1.

The goal of automated discovery of innovative solutions is approached by strenghtening the explorative behavior in the search induced by the CMA-ES. Exploration schemes compatible with this algorithm, but not limited to it, will be examined, including schemes that use a *surrogate model* to determine a path through the search space. In this usage, though, technically speaking, it is not a surrogate model: While the model approximates the quality function, it is not intended to replace the quality function, but to operate complementary to it, for inducing exploration. Nevertheless, the meaning of a model that approximates the quality function stands, and therefore this term will be used throughout the thesis.

Two techniques for approximation modeling, *feed-forward neural networks* and *Kriging*, are described in Section 2.2. Next, to distinguish the explorative usage of surrogate models from their default application in partially replacing the quality function in optimization, we briefly summarize *surrogate-assisted optimization* in Section 2.3. Moreover, related to surrogate-assisted optimization, ideas of how to deal with the trade-off between exploitation and exploration are addressed.

2.1 · Evolutionary Algorithms

Inspired by Darwinian evolution, *Evolutionary Algorithms* (EAs) (see, e.g., [**Bäck, 1996, Eiben and Smith, 2003**]) are *optimization algorithms* that work from a *population* of solutions, using randomness to generate new solutions as slightly-varied combinations of solutions currently in the population. An optimization algorithm aims to find a solution that, given a certain *quality function*, provides the best quality value from the range of that quality function. As such, in EAs, a solution's quality value is used as indicator of *fitness*, scaling its chances in participating in generating new solutions and proceeding into the next *generation*'s population of solutions.

A solution is a possible input to the quality function, where both input and quality function are defined in the broadest sense of the term. It is not uncommon that a different solution representation than the actual input is used *within* the EA, provided that there is an approach to obtain the function input from this *encoded individual*. Typically, an individual in an EA is a vector of values, discrete or continuous, and possibly a combination of these, although other representations such as complete program trees are used as well [**Eiben and Smith, 2003**]. Furthermore, EAs have flexibility in how new solutions are generated, allowing for problem-specific *variation operators*.

An EA performs stochastic sampling, directed by quality via a population of solutions. In its search for the global optimum, being directed by a population of solutions provides moderate robustness to multimodality in the quality landscape [**Loshchilov et al., 2012**] (i.e., the presence of multiple peaks instead of a single increasing or decreasing trend towards the global optimum). Moreover, an EA is able to operate on *black-box* functions [**Kruisselbrink, 2012**], in which the structure of the quality function is not available to the optimization algorithm, for instance when the quality value is determined through simulation. Also, the stochasticity involved makes that not a fixed path is followed through the search space, potentially allowing for repeated execution from the same initial settings to produce alternative solutions.

Evolution Strategies (ESs), constituting a sophisticated class of EAs, feature online adaptations in their probabilistic sampling of new solutions during the search. Typically using a real-valued solution representation, after each iteration, the continuous search distribution from which solutions are sampled is adapted to better meet the search objective. Among ESs, the *Covariance Matrix Adaptation Evolution*

Strategy (CMA-ES) [**Hansen and Ostermeier, 2001**] has proven a successful, widely used variant, in which the dynamic search distribution is an anisotropic, multivariate normal distribution.

2.1.1 · Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

The CMA-ES estimates a covariance matrix to match the local curvature of the quality landscape [**Kruisselbrink, 2012**], but without using or approximating gradient information[1]. For adapting the covariance matrix, the *ranking* of sampled solutions that is induced by the quality function is relied on, instead of the *actual* structure of the quality landscape. As a result, the CMA-ES is applicable to problems on which the usually faster, classical optimization approaches such as *quasi-Newton* and *conjugate gradient* methods fail due to the quality landscape not being convex or being locally highly rugged [**Hansen, 2011**].

We apply the (μ, λ)-CMA-ES, as implemented in the Shark Machine Learning Library v2.3.43[2], thereby following the procedure and parameter settings described in [**Hansen and Kern, 2004**]. The prefix (μ, λ) indicates a parent population size of μ individuals and an offspring population size of λ individuals, and the use of *comma-selection*: At the end of each generation, all parents are discarded and a new set of parents is selected from the lastly-generated offspring. An initial comparison of the setup of [**Hansen and Kern, 2004**] to the updated procedure and parameter settings described in [**Hansen, 2011**] was performed on the real-world airfoil test case described in Chapter 6. It showed that by inducing less locally-oriented search, the original settings delivered better results for this application.

An overview is given of the inner workings of the (μ, λ)-CMA-ES, see Technical Note 2.1, together with the parameter settings adopted for it, see Technical Note 2.2, as used in the experiments reported on in this thesis. Utilizing the (μ, λ)-CMA-ES requires a problem-specific setting for the initial solution from which to start the optimization, to provide an initial global stepsize for the search distribution, and to define termination criteria, for instance, the number of generations to use for the optimization run. For assisting in choosing the parent and offspring population sizes μ and λ, a minimal setting for λ and default setting for μ are provided for the (μ, λ)-CMA-ES, see Equation 2.10.

[1] According to Nikolaus Hansen, on `https://www.lri.fr/~hansen/cmaesintro.html`.
[2] `http://shark-project.sourceforge.net`

Technical Note 2.1 The (μ, λ) CMA Evolution Strategy (CMA-ES)

Initialization:

- Initialize evolution path, $\mathbf{p}_\sigma = \mathbf{0}$ and $\mathbf{p}_\mathrm{c} = \mathbf{0}$, and covariance matrix, $\boldsymbol{C} = \mathbf{I}_n$;
- Problem-specifically, initialize the distribution mean $\mathbf{m} \in \mathbb{R}^n$ (i.e., set to the initial solution) and stepsize $\sigma \in \mathbb{R}_+$, where n is the dimension of the solution vectors.

Until termination:

- Sample λ new individuals $\mathbf{x}_k$:

$$\mathbf{z}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n), \tag{2.1}$$

$$\mathbf{y}_k = \boldsymbol{C}^{\frac{1}{2}}\mathbf{z}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{C}), \tag{2.2}$$

$$\mathbf{x}_k = \mathbf{m} + \sigma\mathbf{y}_k \sim \mathcal{N}(\mathbf{m}, \sigma^2\boldsymbol{C}); \tag{2.3}$$

- Select the μ best-ranked individuals with index $i{:}\lambda$, $i = 1, \ldots, \mu$ and recombine them using weights w_i to form the new distribution mean $\mathbf{m}$:

$$\mathbf{m} \leftarrow \langle \mathbf{x} \rangle_\mathrm{w} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}, \tag{2.4}$$

$$\langle \mathbf{y} \rangle_\mathrm{w} = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}; \tag{2.5}$$

- Update the evolution path, a memory of the steps taken by the population, using exponential smoothing with different update rates for the stepsize and the covariance matrix:

$$\mathbf{p}_\sigma = (1 - c_\sigma) \cdot \mathbf{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma) \cdot \mu_\mathrm{eff}} \cdot \langle \mathbf{y} \rangle_\mathrm{w}, \tag{2.6}$$

$$\mathbf{p}_\mathrm{c} = (1 - c_\mathrm{c}) \cdot \mathbf{p}_\mathrm{c} + \sqrt{c_\mathrm{c}(2 - c_\mathrm{c}) \cdot \mu_\mathrm{eff}} \cdot \boldsymbol{C}^{-\frac{1}{2}} \cdot \langle \mathbf{y} \rangle_\mathrm{w}; \tag{2.7}$$

- Update the stepsize, comparing the length $\|\mathbf{p}_\sigma\|$ of the evolution path to the expected length $\mathbf{E}[\|\mathcal{N}(\mathbf{0}, \mathbf{I}_n)\|]$ under random selection:

$$\sigma \leftarrow \sigma \cdot \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma\|}{\mathbf{E}[\|\mathcal{N}(\mathbf{0}, \mathbf{I}_n)\|]} - 1\right)\right); \tag{2.8}$$

- Update the covariance matrix, by exponential smoothing, in the direction of evolution path $\mathbf{p}_\mathrm{c}$, with the covariances used for obtaining the best μ individuals:

$$\boldsymbol{C} \leftarrow (1 - c_\mathrm{cov}) \cdot \boldsymbol{C} + c_1 \cdot \mathbf{p}_\mathrm{c}\mathbf{p}_\mathrm{c}^\mathrm{T} + c_\mu \cdot \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}\mathbf{y}_{i:\lambda}^\mathrm{T}. \tag{2.9}$$

Technical Note 2.2 Parameter Settings of the CMA-ES

Minimal settings for population sizes μ and λ [**Hansen and Kern, 2004**], where n is the dimension of the solution vectors:

$$\lambda = 4 + \lfloor 3 \ln n \rfloor, \quad \mu = \left\lfloor \frac{\lambda}{2} \right\rfloor; \tag{2.10}$$

Recombination weights (deviating slightly from [**Hansen and Kern, 2004**], cf. $\ln(i)$):

$$w_i = \frac{w_i'}{\sum_{j=1}^{\mu} w_j'}, \quad w_i' = \ln(\mu + 1) - \ln(1 + i), \quad i = 1, \ldots, \mu; \tag{2.11}$$

The *effective mass* μ_{eff} after selection and recombination using weights w_i:

$$\mu_{\text{eff}} = \left(\sum_{i=1}^{\mu} w_i^2 \right)^{-1}, \text{ since } \sum_{i=1}^{\mu} w_i = 1, \text{ it holds that } 1 \leq \mu_{\text{eff}} \leq \mu; \tag{2.12}$$

The evolution-path update rate c_σ for the stepsize and c_{c} for the covariance matrix:

$$c_\sigma = \frac{\mu_{\text{eff}} + 2}{n + \mu_{\text{eff}} + 3}, \quad c_{\text{c}} = \frac{4}{n + 4}; \tag{2.13}$$

The stepsize-update dampening parameter d_σ, decreasing the effect of update rate c_σ:

$$d_\sigma = 1 + 2 \cdot \max \left(0, \sqrt{\frac{\mu_{\text{eff}} - 1}{n + 1}} - 1 \right) + c_\sigma; \tag{2.14}$$

The covariance-matrix update rate c_{cov}, and the influence of the evolution path and of the covariances in each update:

$$c_{\text{cov}} = \frac{1}{\mu_{\text{eff}}} \frac{2}{\left(n + \sqrt{2}\right)^2} + \left(1 - \frac{1}{\mu_{\text{eff}}}\right) \min \left(1, \frac{2\mu_{\text{eff}} - 1}{(n + 2)^2 + \mu_{\text{eff}}} \right), \tag{2.15}$$

$$c_1 = c_{\text{cov}} \frac{1}{\mu_{\text{eff}}}, \quad c_\mu = c_{\text{cov}} \left(1 - \frac{1}{\mu_{\text{eff}}}\right). \tag{2.16}$$

After setting the problem-specific parameters and initializing the covariance matrix to the n-by-n identity matrix, optimization is started, divided into iterations in which five recurring steps are performed:

1. λ offspring individuals are generated based on the current distribution mean $\mathbf{m}$,

covariance matrix $\boldsymbol{C}$, and global stepsize σ;

2. From these individuals, the μ best-ranked individuals are selected and recombined into the new distribution mean, with more influence for the better ranked solutions, see Equation 2.11. Notation $i:\lambda$ stands for the i-th best-ranked individual from the λ total;

3. The evolution-path variables are updated, separately for use later in the stepsize and covariance matrix adaptation. Through exponential smoothing, these variables represent a weighted average of the last search steps performed by the optimization. Here μ_{eff} indicates the resulting *effective mass* of applying the weighted recombination, in terms of the amount of individuals that the recombined mean represents, see Equation 2.12. In the covariance-matrix evolution-path update rule, $\boldsymbol{C}^{-\frac{1}{2}}$ rescales the last step taken based on the width of the search distribution per dimension, to make the length of the step independent of its direction (the H_σ parameter, enforcing a maximum for the last step taken in [**Hansen and Kern, 2004**], is not used);

4. The stepsize is updated, by comparing the length of the evolution path, i.e., the exponentially-weighted average of the last steps taken, to the expected length of a random step. When the length of the evolution path is larger than the expected length under random selection, the stepsize is increased, and the stepsize is decreased when the observed length is smaller than the expected length (for an illustrated motivation, see [**Hansen, 2011**]); $\mathbf{E}[\|\mathcal{N}(\mathbf{0},\mathbf{I}_n)\|]$ is approximated as $\sqrt{n}\left(1-\frac{1}{4n}-\frac{1}{21n^2}\right)$ [**Hansen and Kern, 2004**];

5. The covariance matrix is updated, also using exponential smoothing, based on the exponentially-weighted average of the last steps taken, and the covariances that were used in obtaining the new distribution mean.

Optimization continues until a termination criterion is met. For a discussion of the parameters listed in Technical Note 2.2 and the derivation of possible settings, see [**Hansen and Ostermeier, 2001**].

2.2 · Approximation Models

An *approximation model* predicts a *mapping* from inputs to outputs by *generalizing* from a *training set* of inputs with corresponding outputs, to approximated outputs for inputs not encountered in training [**Haykin, 1998**]. As such, it represents an approximation of the actual function from input to output. The inputs are vectors that are associated with singular or higher-dimensional output vectors. Both type of vectors can consist of discrete or continuous variables, or a combination of both. For employing approximation modeling, a data set of inputs mapped to outputs is required. To then predict outputs for unseen inputs, a *modeling technique* is to be chosen, as well as an *iterative training approach*, which fits the model to the available data guided by an *error measure* that indicates the attained generalization performance.

Two modeling techniques are presented, *feed-forward neural networks* and *Kriging*. Whereas Kriging is a good *interpolator*, i.e., in general delivers good predictions for points located between training points, and provides an estimate of the associated prediction error, neural networks require less computational effort for training and do usually not converge to a single output vector in *extrapolating*, i.e., predicting outputs for points located outside of the space covered by the training data.

Feed-forward Neural Networks

Artificial neural networks (ANNs) (e.g., see [**Haykin, 1998**]) are inspired from neural networks in animals and consist of interconnected nodes or *neurons*. A neuron is a signal-propagating unit that sends a signal over its outgoing connections if its input signal, i.e., the total strength of the signals received on its incoming connections, exceeds a certain *threshold*. In using ANNs for approximation modeling, the strengths or *weights* of the connections between nodes are adapted in fitting the model to training data. The threshold is implemented as an *activation function* that transforms the sum of the weighted incoming values to an output value, either via a discrete step function or as a transformed continuous output value.

In an ANN, nodes can be grouped in *layers*, with connections only existing between nodes in adjacent layers. *Feed-forward neural networks* (FFNNs) are defined to have signals propagate forward between subsequent layers. They consist of a layer of input nodes and a layer of output nodes, and typically have one or more *hidden layers* in between, in which case they are termed *multilayer perceptrons* [**Haykin,**

1998]. The input nodes have linear activation, i.e., they just propagate the actual values of the input vectors from the training set to the nodes in the first hidden layer.

Multilayer perceptrons are trained using *error back-propagation* [Rumelhart et al., 1986, Haykin, 1998]: The prediction error between the FFNN-approximated output and the actual output of a training example is propagated back through the network, slightly adjusting the weights along the way to make the approximated output closer to the desired output. Training is continued in steps that feed the whole training set to the network, i.e., one *epoch*, while back-propagating the prediction error for each training example, until a required precision that is expressed by an error measure, or an available number of epochs is reached.

In the experiments described in this thesis, we apply FFNNs as implemented in the Shark Machine Learning Library v2.3.43[3]. Nodes in the hidden layer are sigmoidally-activated, i.e., they have a continuous *logistic* activation function [Haykin, 1998], while the output nodes have linear activation. The network is fully connected, meaning that there are connections going from each node in a certain layer to all nodes in all following layers. A *bias node*, i.e., always producing a signal with value 1.0, is connected to all nodes, except those in the input layer. For training, the iRprop$^+$ algorithm by [Igel and Hüsken, 2003] is applied, an improved version of the *resilient back-propagation* algorithm that uses an adaptive update scheme for the connection weights, based on the sequence of recent changes for a certain weight [Riedmiller and Braun, 1993].

Kriging

In *Kriging*, a technique originally intended for predicting ore concentrations in mining, approximation modeling is performed based on a *Gaussian random field* [Emmerich, 2005]. A Gaussian random field defines a *separate* Gaussian distribution at each possible input point, i.e., *index* of the random field, in a given domain. A to-be-modeled function with one-dimensional output is assumed to be a *realization* of a Gaussian random field. This enables using the Gaussian distribution at each index for modeling the associated output value.

The procedure is illustrated in Figure 2.1. The training points, for which the actual output value is available, are represented by the white dots. Assuming spatial correlation between these output values [Jones et al., 1998, Emmerich, 2005], an

[3] http://shark-project.sourceforge.net

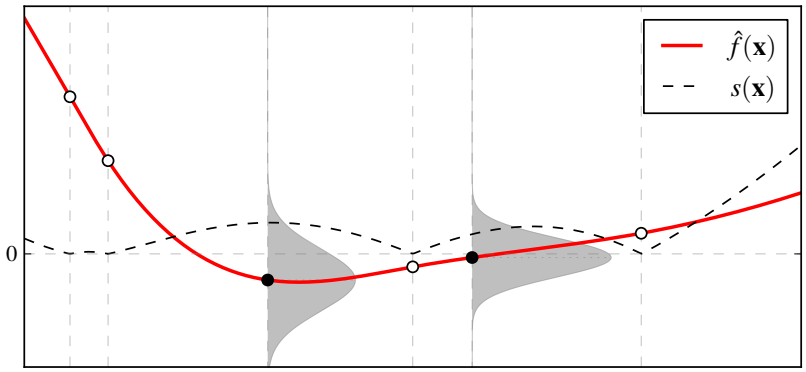


Figure 2.1 Kriging. The underlying output function of training points (white dots) is taken as a realization of a Gaussian random field (i.e., a random field defining Gaussian distributions in each input $\mathbf{x}$). By assuming spatial correlation between the training points, an approximation $\hat{f}(\mathbf{x})$ (red line) of the underlying function is calculated. Together with the prediction error $s(\mathbf{x})$ (black dashed line) that is estimated for each input as well, this provides the likeliness (via the probability density function of $\mathcal{N}\left(\hat{f}(\mathbf{x}), s^2(\mathbf{x})\right)$, in gray) of all possible output values of an unseen input $\mathbf{x}$ (black dots), where $\hat{f}(\mathbf{x})$ is the most-likely realization of the output value.

approximation $\hat{f}(\mathbf{x})$ is calculated of the to-be-modeled function, plotted as a red line. This red line indicates the mean of the Gaussian distribution at each index (i.e., input point) in the random field. The standard deviation $s(\mathbf{x})$ per Gaussian distribution, termed the *conditional* or *local* standard deviation, is also determined from the assumed correlation between the training points and their spatial distribution, and is plotted as a black dashed line. Altogether, for an unseen input point $\mathbf{x}$, this provides the likelihood for a certain value to be the actual output value, as it is considered to be a realization of the output distribution $\mathcal{N}\left(\hat{f}(\mathbf{x}), s^2(\mathbf{x})\right)$. Two unseen input points are indicated by the black dots, with the plotted probability density functions showing the likelihood of all their possible output values.

Kriging is not a branch of *Gaussian Process modeling* (GP) [**Rasmussen and Williams, 2006**] as it is often presented: Kriging aims to predict the output of an unseen input point as a linearly-weighted sum of the outputs in the training data, based on the spatial relationship between the training data and the unseen input point (i.e., the *best linear unbiased predictor* [**Stein, 1999**]). GP, on the other hand, aims to provide the mean and conditional standard deviation of the Gaussian distribution of the output of an unseen input point. Nevertheless, we restrict Kriging to using Gaussian-type random fields and therefore both interpretations can be used interchangeably, as

is done in the previous paragraph. In case of multiple output variables, i.e., the to-be-modeled function has multiple outputs, multiple separate instances of univariate Kriging, described above, can be used. This, however, ignores the correlations between the output variables, which can be taken into account by using multivariate Kriging (see, e.g., [**Kleijnen and Mehdad, 2012**]).

In this work *Ordinary Kriging* is employed, which assumes a constant trend ("offset") for the Gaussian random field. The spatial correlation is represented by a power-exponential multiparameter kernel [**Forrester et al., 2008**],

$$\mathrm{corr}(\mathbf{x},\mathbf{x}') = \exp\left(-\sum_{i=1}^{\dim(\mathbf{x})} \theta_i |x_i - x_i'|^{p_i}\right), \tag{2.17}$$

$$\theta_i \in [10^{-10}, 10^2],\ p_i \in [1,2], \tag{2.18}$$

where $\dim(\mathbf{x})$ is the number of elements in vector $\mathbf{x}$ and where the θ_i and p_i are allowed to vary per dimension. The Gaussian random field is trained by tuning the correlation function: The parameters θ_i and p_i of the correlation function are chosen such that the likelihood that the available outputs were sampled from the Gaussian random field is maximized. Finding settings for the θ_i and p_i is done using *Differential Evolution* [**Storn and Price, 1997**] with population size 50, for 50 generations, followed by a hill-climbing phase. The used Kriging implementation was provided by Giles Endicott, implemented according to [**Forrester et al., 2008**].

2.3 · Surrogate-assisted Optimization

In *surrogate-assisted optimization*, approximation models are employed for improving efficiency in terms of the number of consumed quality evaluations by using model approximations as a *surrogate* indication of quality. In this usage, approximation models are hence addressed as *surrogate models*, or as *metamodels*, the latter referring to the fact that the quality function that is approximated can be seen as a model in its own right, namely, of actual real-world usefulness or applicability of a solution. In an attempt to more extensively utilize the information implicitly available from the sampled data, a surrogate model is trained on the solutions that were evaluated on quality during optimization, and its quality predictions are used for ranking solutions. Potentially, this way, less evaluations of the quality function are required for the same performance as in an optimization scheme that is not model-assisted, either enabling

to decrease the number of evaluations necessary for obtaining comparable results, or to find better solutions under the same evaluation budget.

In replacing quality function evaluations by model approximations, one should bear in mind that models are susceptible to introducing false and obscuring actual high-quality regions, depending on the interplay between the regularity of the quality landscape, the number and distribution of the training points, and the used (settings for the) modeling technique. Therefore, it is important to keep evaluating new solutions on actual quality during the optimization process for verification of the followed path, either via presenting the new information to the optimizer directly [**Jin et al., 2002, Emmerich et al., 2006**], followed by a model update (more "local" use of the model), or through restarting the search on an updated model [**Jones et al., 1998, Sóbester et al., 2004**] (more "global" use of the model). Selecting the most-promising solutions for on-line evaluation should be guided by the model predictions of the quality *and* by information on the distribution of the training points. This way, areas with relatively-low predicted quality, while nevertheless likely to harbor high-quality solutions, are considered as well, decreasing the tendency to skip over potential high-quality regions due to lack of information.

Next to the predicted quality of a solution, certain modeling techniques therefore provide the *uncertainty* associated with the prediction based on the distance to the available training points (e.g., Kriging, see Section 2.2, and *radial basis function networks* (RBFNs) [**Sóbester et al., 2004**]). The uncertainty of a predicted value $\hat{f}(\mathbf{x})$ for a solution $\mathbf{x}$ is provided as the standard deviation $s(\mathbf{x})$ of a normal distribution $\mathcal{N}\left(\hat{f}(\mathbf{x}), s^2(\mathbf{x})\right)$ around the predicted value $\hat{f}(\mathbf{x})$. The actual quality value $f(\mathbf{x})$ will be a realization of a random variable that is distributed according to $\mathcal{N}\left(\hat{f}(\mathbf{x}), s^2(\mathbf{x})\right)$ [**Jones et al., 1998**].

2.3.1 · Surrogate Utility Functions

The most-straightforward utility function is applying the approximated quality function $\hat{f}$ for ranking solutions directly, thus relying on the predicted quality values without considering uncertainty, i.e., for a quality function f that is to be minimized,

$$\operatorname*{argmin}_{\mathbf{x}} \hat{f}(\mathbf{x}). \tag{2.19}$$

According to [**Kushner, 1962**], the purpose of a utility function is to introduce a trade-off between sampling in expected high-quality regions versus sampling in

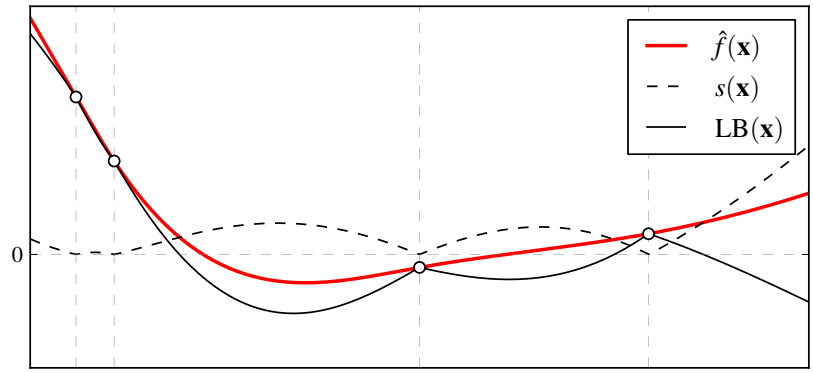


Figure 2.2 Lower Confidence Bound. A surrogate model is used that, fitted to training data of a to-be-minimized quality function, represented by white dots, provides both a quality prediction $\hat{f}(\mathbf{x})$ and an estimated associated prediction error $s(\mathbf{x})$ for unseen solutions $\mathbf{x}$. In minimizing LB$(\mathbf{x})$, here weighting both prediction and error equally, i.e., $w = 1$, solutions in the far-right corner get a good ranking, even though their expected quality values $\hat{f}(\mathbf{x})$ are suboptimal.

under-explored regions [**Emmerich, 2005**]. In Equation 2.19 the balance is tilted completely towards the first, selecting directly on the expected quality values. An intuitive ranking criterion taking both into account, as the predicted quality value $\hat{f}(\mathbf{x})$ and the associated uncertainty $s(\mathbf{x})$ are of the same scale, is combining them using a linear weighting. This results, for a to-be-minimized quality function, in ranking on the *lower confidence bound* (LB) [**Cox and John, 1997**, **Emmerich et al., 2006**, **Emmerich, 2005**],

$$\underset{\mathbf{x}}{\operatorname{argmin}}\, \mathrm{LB}(\mathbf{x}), \quad \mathrm{LB}(\mathbf{x}) = \hat{f}(\mathbf{x}) - w \cdot s(\mathbf{x}), \quad w > 0. \tag{2.20}$$

A lower w-value makes that there is more emphasis on exploitation of the expected high-quality regions, while a higher w-value drives the search towards areas with higher uncertainty, those areas that have been sampled less densely. Taking $w = 1$, the LB criterion is illustrated in Figure 2.2.

The most-frequently encountered utility function in literature that takes the prediction $\hat{f}(\mathbf{x})$ and associated error $s(\mathbf{x})$ into account, is the *expected improvement* (ExI) criterion. ExI quantifies how much an unseen solution's quality value is expected to improve upon the currently best-found quality value. It does so based on probability distributions that are parameterized according to these predictions and errors, e.g., Gaussian distributions $\mathcal{N}\left(\hat{f}(\mathbf{x}), s^2(\mathbf{x})\right)$, depending on what probability distributions are assumed in the used modeling technique.

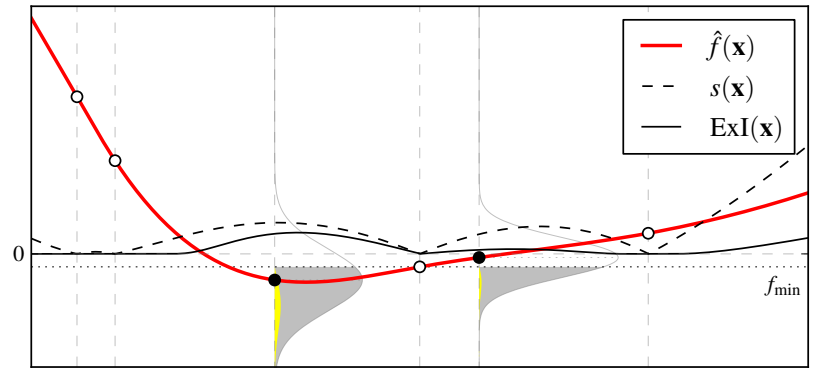


Figure 2.3 Expected Improvement. Based on prediction $\hat{f}(\mathbf{x})$ and estimated prediction error $s(\mathbf{x})$, the improvement that is expected to be realized in solution $\mathbf{x}$, with respect to $f_{\min}$, is calculated by summing over all possible improvements in $\mathbf{x}$ times their probability. Training points are indicated by white dots, $f_{\min}$ is the quality value of the best-quality training point. For two unseen solutions, indicated by black dots, the expected improvement ExI($\mathbf{x}$) is depicted by the size of the yellow surface.

Ranking solutions on expected improvement can be done using the ExI($\mathbf{x}$) utility function [**Mockus et al., 1978, Emmerich, 2005**],

$$\underset{\mathbf{x}}{\operatorname{argmax}}\, \mathrm{ExI}(\mathbf{x}), \quad \mathrm{ExI}(\mathbf{x}) = \int_{-\infty}^{f_{\min}} (f_{\min} - y) \cdot \mathrm{pdf}_{\mathcal{N}\left(\hat{f}(\mathbf{x}), s^2(\mathbf{x})\right)}(y)\, \mathrm{d}y, \tag{2.21}$$

$$= \int_{-\infty}^{f_{\min}} (f_{\min} - y) \cdot \mathrm{pdf}_{\mathcal{N}(0,1)}\left(\frac{y - \hat{f}(\mathbf{x})}{s(\mathbf{x})}\right) \mathrm{d}y. \tag{2.22}$$

For a candidate solution $\mathbf{x}$, all possible improvements with respect to the best-found quality value $f_{\min}$ are weighted by their probability, and summed-up. The possible improvements can be represented by the part of a vertical line through $\mathbf{x}$ that extends under the line $f_{\min}$, see Figure 2.3. The probabilities of these improvements are then indicated by the height of the probability density function at the same vertical position. Note that it is the standard deviation $s(\mathbf{x})$ that appears in the Equation 2.22, not variance $s^2(\mathbf{x})$. The expression can be simplified [**Jones et al., 1998, Emmerich, 2005**] to

$$\mathrm{ExI}(\mathbf{x}) = \left(f_{\min} - \hat{f}(\mathbf{x})\right) \cdot \mathrm{CDF}_{\mathcal{N}(0,1)}\left(\frac{f_{\min} - \hat{f}(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x}) \cdot \mathrm{pdf}_{\mathcal{N}(0,1)}\left(\frac{f_{\min} - \hat{f}(\mathbf{x})}{s(\mathbf{x})}\right). \tag{2.23}$$

The implicit trade-off that is present in ranking on ExI, i.e., between sampling expected high-quality regions and under-explored regions, depends on the ruggedness that is estimated from the training points. If this ruggedness is estimated greater than

the actual irregularity in the quality landscape, time might be wasted on solutions with suboptimal quality, as the errors $s(\mathbf{x})$ are too large, and thereby sampling in low-quality regions is encouraged. On the other hand, if the ruggedness is estimated smaller than the actual irregularity, this can cause the global optimum to be ignored.

2.4 · Summary

This chapter introduced techniques that will be used in the remainder of the thesis, either applying them directly or using them as inspiration, with the goal of assisting automated search in targeting *innovative solutions*. In applying the *CMA-ES*, an iterated adaptive search method used as underlying optimizer in all experiments, a problem-specific initial mean (i.e., initial solution) and stepsize are to be set for its adaptive search distribution.

Feed-forward neural networks (FFNNs) and *Kriging*, two approximation-modeling techniques, differ from each other in the sense that FFNNs are computationally less expensive to apply, and hence better applicable in scenarios with high-dimensional inputs or a large amount of training data. While Kriging is better at interpolating points located in-between training data, it converges to the average output value of the training points in extrapolating.

Surrogate-assisted optimization replaces the quality function by a surrogate model of it in ranking candidate solutions. During surrogate-assisted optimization, it is important to keep evaluating solutions on actual quality to verify the search path, because of possible modeling errors. Certain surrogate utility functions, such as the *expected improvement* (ExI), account for estimated modeling errors by including them in calculating the expected quality value of solutions.

3

Innovativeness in Design Optimization

Finding *innovative solutions* is an objective that holds generally for design optimization tasks, as when it is not aimed for directly, it can at least be stated to be valued in general. To be able to assist human engineers in finding innovative solutions using automated search, what exactly constitutes an innovative solution has to be determined.

[**Garcia and Calantone, 2002**] state in their survey on innovation terminology that, while often used interchangeably, it is important to make a distinction between the constructs *innovation* and *innovativeness* (i.e., the property of being innovative): An innovative solution becomes an innovation once it has passed through production and marketing phases and is diffused into the marketplace. Factors accounting for a swift diffusion of such an invention into society are addressed in [**Rogers, 2010**]. Furthermore, [**Garcia and Calantone, 2002**] stress to clearly distinguish between innovativeness relating to *products* and *organizational* innovativeness: Organizational innovativeness involves the *tendency* of an organization to develop or adopt innovative products, i.e., lead to or participate in an innovation, while the innovativeness of a product that a firm markets or adopts *cannot* be regarded as a measure of organizational innovativeness. In this thesis, innovativeness is addressed from its viewpoint of being a property of technological products.

[**Deb and Srinivasan, 2006**] describe the practice of using automated search for uncovering innovative solutions. Their approach, termed *innovization* (e.g., [**Deb and Srinivasan, 2006, Deb, 2013**]), is reformulating optimization problems involving a single search objective as a *multiobjective* problem with two or more conflicting search

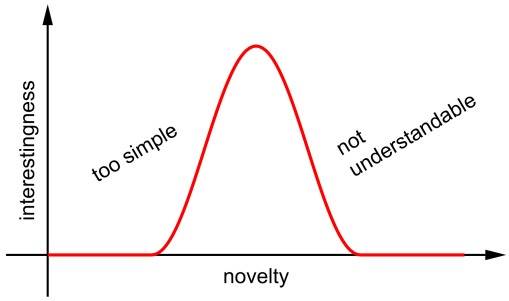


Figure 3.1 Novelty vs. Interestingness. Interestingness shown as a function of novelty. Image courtesy of [**Graziano et al., 2011**].

objectives, for instance utility and cost. In optimizing on these objectives in parallel, the aim is to best-approximate the optimal multiobjective set of solutions that all have a different trade-off in objective values: Of each such a solution cannot be stated that another solution in the set is better with respect to all objectives. This approximated *non-dominated* set is then analyzed to find common design principles shared by the different solutions in the set. Potentially, this uncovers common solution principles, new to the human engineer, that lead to high-performing designs. Thus, innovization is an approach of using automated search for gaining a deeper understanding of the optimization problem at hand, potentially offering new insights that *can* lead to innovative products.

For a definition that allows for quantifying innovativeness, we turn to the discussion on *creative* products in [**Saunders, 2002**]. Generally, creative products are recognized as being *useful* and *novel*. Usefulness has to do with being appropriate, valuable, or aesthetic, while novelty is related to originality, unexpectedness, and surprisingness. Furthermore, a creative product is likely to remain useful for some time, whereas its novelty dissipates much quicker [**Saunders, 2002**]. We take innovativeness as synonymous to creativity in this context, and thus depending on utility and the more volatile novelty.

Other work describing novelty of solutions contrast it to the actual *interestingness* of a solution, related to learning and model building. Novelty of an observation, action, or solution, is a subjective notion in the sense that it depends on *similarity* to what was encountered before, and on the *state* of the agent, approach, or model used in *determining* this similarity. Easily modeled and frequently occurring solutions quickly lose their novelty, while solutions that are subject to *noise* always remain novel

Table 3.1 Terminology. Overview of the terms that are used in defining *innovative solutions* and their likelihood of being adopted by the engineering community, to get further-developed into an innovative product.

Term	Description
Solution	Product proposal, product design.
State-of-the-art	Highest-quality solutions, generally accepted as such, in the application domain.
Innovation	The process of an innovative product getting adopted by the customer base, i.e., the end users.
Innovative, innovativeness	1) *Novel* and 2) of *better or tolerably worse quality* compared to 3) the *state-of-the-art*; synonymous to creative.
Novel, novelty	Difference with respect to a *reference set*, expressed using a *domain-specific distance measure*.
Interestingness	Increase of understanding that is given rise to, with respect to a *reference model*; corresponds to: Sufficient, but not too extreme novelty.

[**Graziano et al., 2011**]. For efficient learning or modeling of a solution space, those solutions are selected at each time step that improve the understanding or model of the solution space most: An interesting solution has sufficient (not too simple, boring) but not too extreme novelty (beyond understandable, for instance, noise) [**Schmidhuber, 1997**]. In learning, not the most-novel but the most-interesting solutions are to be selected, see Figure 3.1.

In the sections that follow, we further relate these notions to the aim of using automated search to assist in finding innovative solutions. We start by defining novelty, innovativeness, and interestingness in Section 3.1. An overview of the terminology that will be used is given in Table 3.1. Next, in Section 3.2, it is concluded that finding innovative solutions is best done indirectly through aiming for diversity. Possible approaches for efficiently searching for diverse solutions, inspired by the discussion on novelty and interestingness, are described in Section 3.3. The chapter is summarized in Section 3.4.

3.1 · Novelty, Innovativeness, and Interestingness

The aim of this thesis is to develop an automated search method that assists a human engineer in finding *innovative solutions*. Here, a *solution* is to be viewed as a *product proposal* or product design, to be further developed into an actual product by a human domain expert. Thus, *product innovativeness* is meant, related to technological products, and not *organizational* innovativeness from a business perspective (see [**Garcia and Calantone, 2002**]). An innovative product, i.e., developed

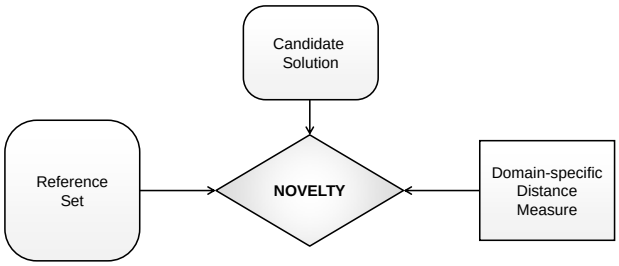


Figure 3.2 Novelty. The novelty of a candidate solution is expressed with respect to a reference set of solutions, and is quantified as the domain-specific distance to the reference solution that is closest to the candidate solution.

from an innovative solution, has the potential of leading to an *innovation*, which is roughly defined as an innovative product attaining market penetration [**Garcia and Calantone, 2002**]. A solution's innovativeness should thus quantify this potential for the product that the solution gives rise to.

To express innovativeness of a solution, *novelty* of a solution is introduced first. We define novelty as a property that is expressed relative to a *comparison* or *reference set* of solutions, by applying a *distance measure* appropriate for the application domain of the solutions. Such a distance measure is required to highlight those aspects of the resulting product that are relevant from the application perspective (e.g., depending on the application, the color of an automobile is not, or actually might be relevant). This is formalized as follows, see also Figure 3.2.

Notion 3.1 Novelty. *The novelty of a solution* $\mathbf{x}$ *is defined as its distance to solutions* $\mathbf{x}_{\text{ref}}$ *in a reference set* S_{ref}*, with respect to a domain-specific distance measure* d*. A solution is generally represented by a vector of values.*

For quantification, a *novelty measure* is required that derives from the measured domain-specific distances between the candidate solution $\mathbf{x}$ and the solutions $\mathbf{x}_{\text{ref}}$ in the reference set. We adopt the *smallest observed distance* (referred to as *uniqueness* in Chapter 4) as novelty score.

Like novelty, innovativeness is expressed with respect to a reference set of solutions. We define an innovative solution to be of better, or tolerably less quality than the solutions in this reference set. "Tolerably less" means still sufficient for adoption by a human domain expert, relative to the quality of the solutions in the reference set. A solution's innovativeness can then be quantified by its novelty with respect to the solutions in the reference set.

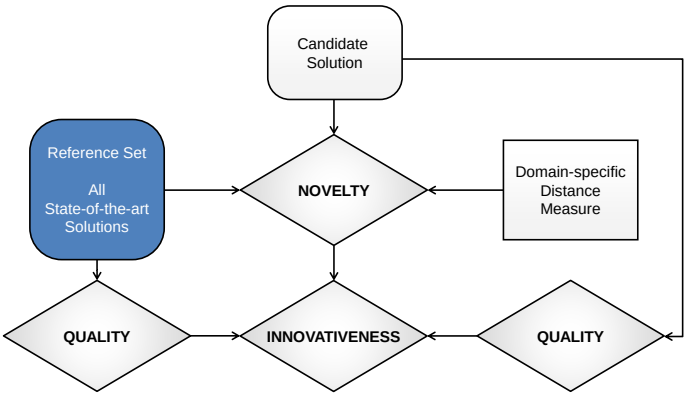


Figure 3.3 Innovativeness. To determine its innovativeness, a candidate solution is compared to the set of state-of-the-art solutions in its application domain. An innovative solution is first of all of better, or tolerably less quality than the most-resembling state-of-the-art solution. Upon meeting this requirement, the innovativeness of a solution is quantified by its novelty, which we define to be equal to the domain-specific distance to this most-resembling solution.

The *state-of-the-art* solutions are the *highest*-quality solutions, generally accepted as such, in an application domain. The meaning of a solution in this sense is that of the product design of an already established high-quality product instance. For concisely expressing innovativeness, the comparison is thus made to these state-of-the-art solutions. When speaking of innovativeness of a solution, it is clear which application domain is meant, namely, of the product that the solution represents. We will therefore simply refer to innovativeness, when innovativeness with respect to the reference set of the state-of-the-art solutions in the application domain is meant. This is formalized as follows, see also Figure 3.3.

Notion 3.2 Innovativeness. *An innovative solution* $\mathbf{x}$ *statisfies a domain-specific required quality level* f_{req}*. The quality level* f_{req} *is derived from the set of best-quality solutions* $S_{\text{state-of-the-art}}$ *in the application domain. Upon meeting this requirement, the innovativeness of a solution* $\mathbf{x}$ *is quantified by its novelty with respect to* $S_{\text{state-of-the-art}}$*.*

The domain-specific required quality level can be formulated as a *relative* value Δf_{req} that puts a restriction on the allowed quality difference to the most-resembling solution $\mathbf{x}_{\text{closest}}$ as follows:

$$|f(\mathbf{x}) - f(\mathbf{x}_{\text{closest}})| < \Delta f_{\text{req}}, \text{ or } f(\mathbf{x}) \text{ is better than } f(\mathbf{x}_{\text{closest}}), \tag{3.1}$$

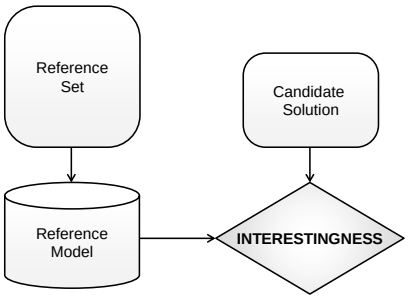


Figure 3.4 Interestingness. The interestingness of a candidate solution is determined with respect to a reference model obtained from reference solutions. Those solutions that represent new knowledge but still resemble the reference solutions (i.e., sufficiently but not extremely novel solutions) are the most-interesting ones, as they typically increase the understanding contained in the reference model most.

where

$$\mathbf{x}_{\text{closest}} = \underset{\mathbf{x}_{\text{state-of-the-art}} \in S_{\text{state-of-the-art}}}{\operatorname{argmin}} \mathrm{d}(\mathbf{x}, \mathbf{x}_{\text{state-of-the-art}}). \tag{3.2}$$

Interestingness

By linearly depending on novelty, innovativeness does not fully express the likelihood of a solution actually being adopted and used by domain experts. This is best captured by the notion of *interestingness* [**Schmidhuber, 1997**]: To be perceived as interesting, a *concept*, understood here in the most-general sense, should be challenging by differing sufficiently from available knowledge, while not deviating so much that it cannot be understood at all. Likewise, an interesting solution should be novel, while still be in line with existing designs [**Saunders and Gero, 2001**].

Interestingness is a highly subjective, application-dependent notion: It depends on one's understanding of the application domain, built upon pre-existing knowledge and experience [**Saunders and Gero, 2001**]. We define a solution to be interesting if it greatly improves the current understanding of the application domain. A closed-form expression of interestingness is not feasible due to the subjectiveness involved. Where novelty is expressed with respect to a reference set, interestingness is related to a *reference model*. By *model*, a *representation of understanding*, i.e., of knowledge, is meant that can be used to generalize from *observed* concepts to as-yet-*unwitnessed* concepts, for instance in mathematical function approximation or as exists in the human mind. This is formalized as follows, see also Figure 3.4.

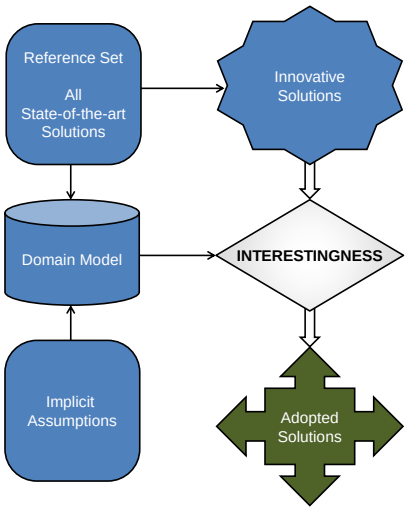


Figure 3.5 Interestingness of Innovative Solutions. Stepping beyond innovativeness, to express the likelihood of a solution actually being adopted by the engineering community, it should be sufficiently novel, but not so novel that it is not fully understood. This interestingness of a solution is expressed with respect to a model of the domain, as exists in a human domain expert.

Notion 3.3 Interestingness. *The interestingness of a solution is the increase of the understanding contained in a reference model that it leads to, after using it to refine the model. A reference model is established from a reference set of solutions.*

Evaluating the interestingness of innovative solutions requires a *domain model* based on the knowledge and understanding of the state-of-the-art solutions in the application domain (see Figure 3.5). Such a model is the comprehension existing in the mind of a human domain expert: The expert finds the candidate solution interesting based on his or her (implicit) knowledge and understanding of the established solutions in the domain. Alternatively, in principle, an artificial model can be used that approximates a certain mapping between solution characteristics, based on a set of established state-of-the-art solutions: By retraining the approximation model on a candidate solution, it is able to predict the to-be-learned mapping better, and this *learning progress* then characterizes the interestingness of a solution. Depending on the type of reference model, and even on the expert that was asked to evaluate, interestingness rankings are likely to show large variation. It is therefore important to describe the exact nature of the used reference model.

A further distinction can be made between interestingness to the *product devel-*

oper, i.e., the human domain expert, and the *product user*, i.e., prospective customers. As the aim of this work is to present methods for automated assistance in finding innovative solutions, ultimately to be developed into actual innovative products, we use the domain-expert perspective of interestingness, which thus expresses the likelihood for a product proposal found in automated search to be adopted by the human engineer and further developed by him or her into an actual product. Notwithstanding, interestingness from the customer's perspective *is* important for the commercial marketing strategy of the further-developed actual product [**Garcia and Calantone, 2002**], namely, it expresses the product's likelihood to be adopted by the potential customer base and thereby formally leading to a *technological innovation* according to the OECD (Organisation for Economic Co-operation and Development) definition [**Freeman, 1991, Garcia and Calantone, 2002**]: *'innovation' is an iterative process initiated by* [...] *a technology-based invention* [...] *striving for the commercial success of the invention*. We take "technology-based invention" synonymous to "innovative product". A theory further explaining the speed of the customer-base's adoption of innovative products is *diffusion of innovations* [**Rogers, 2010**], originally posed in 1962.

Discussion

Recapitulating, our view on innovative solutions is that they are novel and satisfy the minimal-quality requirement for adoption by the engineering community, both derived from the reference set of state-of-the-art solutions in the application domain. The most-innovative solution, however, is not likely to be the most-interesting for an engineer. For instance, when solution quality is based on simulation and the engineer cannot fully understand the simulated physical behavior that the solution gives rise to, he or she will not use it. An argument motivating this is that solutions found through simulation are prone to modeling and/or approximation errors. Hence, an innovative solution is only interesting to an engineer if its innovative properties are "graspable" for him or her. This is analogous to how children learn and play: They are attracted to things that are challenging, while at the same time understandable with respect to their current knowledge and experience.

Not the most-innovative solution, but the most-interesting sufficient-quality solution has the best chance of being selected and further-developed into an actual product. Whereas the most-innovative solution (highest novelty) *in theory* has the best potential of leading to an innovation, i.e., the process of the innovative product that it is further

developed into actually being adopted by the end users, it is the most-interesting innovative solution (sufficient, but not too extreme novelty) that has the best potential of leading to an innovation *in practice*, as it is the most-likely to be adopted by the engineering community and developed into a product.

Notwithstanding, given the intractable nature of interestingness, for finding renewing but not too extreme solutions, the search should be steered towards solutions with maximum innovativeness. The view on interestingness is that it is evaluated in a post-processing step, using a model that is external to the search, which is the general approach of a human domain expert judging produced results after optimization.

We ask ourselves whether an expert could find a solution interesting that was not evaluated as being innovative in our approach. The expert has to agree on the used domain-specific distance measure and on the reference set of state-of-the-art solutions, so he or she should only find solutions interesting that are viewed as novel based on these, and if not, this indicates that distance measure and reference set are to be adjusted. It is possible that low-quality solutions not meeting minimum quality requirements are perceived as interesting, but these then have such optimal novelty (i.e., are located in the appropriate range for inducing learning progress) that the quality tolerance is (implicitly) stretched.

3.2 · Search Objective: Towards Innovative Solutions through Diversity

Searching for innovative solutions requires defining an appropriate domain-specific distance measure that differentiates between solutions. Innovativeness furthermore requires comparing candidate solutions to all established state-of-the-art solutions in the application domain. On the one hand, compiling this comprehensive reference set is hardly realizable, also taking, e.g., historical solutions that were abandoned into account. On the other hand, it is difficult to formulate established, e.g., patented, solutions in such a way that they can be interpreted automatically through putting them in the right notation.

We thus aim for innovative solutions, but it is difficult to express the true innovative value of solutions in an automated way. For certain applications, it could be argued that novelty is expressed with reasonable accuracy through identifying a relatively small number (e.g, 4–5) of baseline instances of established solutions. Alternatively, we drop the requirement of formulating reference solutions altogether. Instead of searching for innovative solutions directly, we aim for solutions that are of *high quality* and

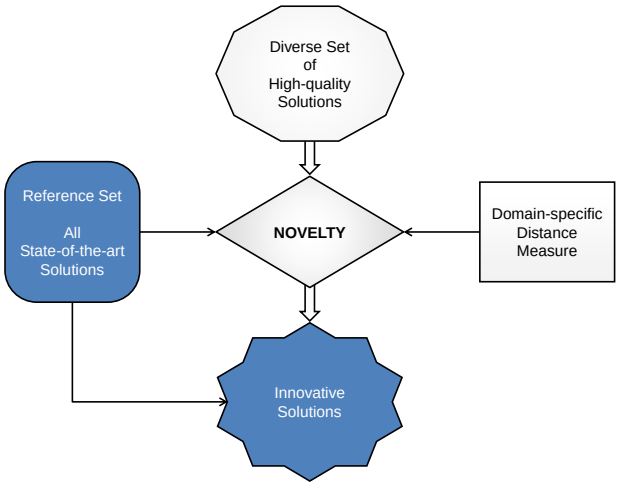


Figure 3.6 Towards Innovative Solutions through Diversity. A diverse set of high-quality solutions potentially contains solutions that are innovative, that is, novel with respect to all established high-quality solutions in the application domain and of better or sufficient quality compared to this state-of-the-art. Diverse means that the solutions are different from each other according to the same domain-specific distance measure as used in novelty determination.

are *diverse* with respect to each other, see Figure 3.6. The idea behind this is that a diverse set of high-quality solutions potentially contains innovative solutions. This chance increases with higher diversity in the set of solutions found.

For determining diversity, defining the domain-specific distance measure is still required, now for comparing candidate solutions with each other. This is formalized as follows, see also Figure 3.7.

Notion 3.4 Diversity. *The diversity of a set S of solutions is expressed by a set diversity measure* D *that applies a domain-specific distance measure* d *to all pairs of solutions* $\mathbf{x}, \mathbf{x}' \in S$.

3.2.1 · Measuring Performance

Resuming, we aim for a search method to produce a diverse set of high-quality solutions, motivated by the potential of innovative solutions being present among them. Therefore, to measure performance of a search method, an approach is required for scoring its result set by reflecting both the quality of the solutions found and the diversity between the solutions in this set.

Postponing a possibly required specification of a minimally-acceptable quality

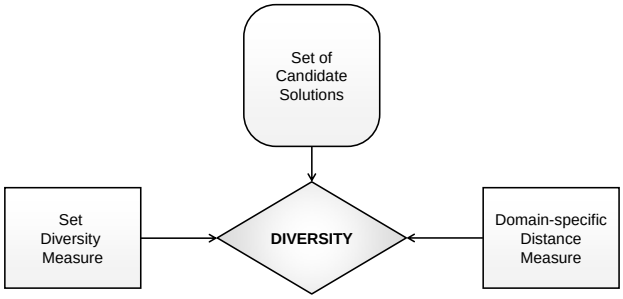


Figure 3.7 Diversity. The diversity of a set of candidate solutions is obtained using a set diversity measure that applies a domain-specific distance measure to all pairs of solutions from the set of candidate solutions.

level, we determine all *level-set* approximations possible in a method's result set, governed naturally by the quality values of the solutions contained in the result set. For each such choice of minimal quality level, the diversity is evaluated of the resulting level-set approximation. Each "solution level set" contains the solution itself and all other found solutions that have quality equal to or better than the solution's quality score. Formally, given a quality function $f\colon X \to \mathbb{R}$ and a threshold quality value f_{thres}, we define a level set as

$$L_{f_{\text{thres}}} = \begin{cases} L^{-}_{f_{\text{thres}}} = \{\mathbf{x} \in X | f(\mathbf{x}) \leq f_{\text{thres}}\} & \text{if } f \text{ is to be minimized} \\ L^{+}_{f_{\text{thres}}} = \{\mathbf{x} \in X | f(\mathbf{x}) \geq f_{\text{thres}}\} & \text{if } f \text{ is to be maximized} \end{cases}, \tag{3.3}$$

that is, the level set $L_{f_{\text{thres}}}$ contains all solutions from domain X with quality equal to or better than f_{thres}.

Set Diversity Measure

Each solution thus gets assigned a diversity score through its quality value serving as f_{thres} for a level-set approximation within the result set. To determine the diversity in a set of solutions S, a *set diversity measure* $\mathrm{D}(S)$ is required that applies the domain-specific distance measure to all pairs of solutions. Following [**Ulrich and Thiele, 2011**], we assume a symmetric domain-specific distance measure,

$$\mathrm{d}(\mathbf{x},\mathbf{x}') = \mathrm{d}(\mathbf{x}',\mathbf{x}) \text{ for any pair of solutions } \mathbf{x},\mathbf{x}'. \tag{3.4}$$

In [**Solow and Polasky, 1994**], three requirements are given for a set diversity measure, structured in [**Ulrich et al., 2010, Ulrich and Thiele, 2011**]:

1. *Monotonicity in variety*: Diversity increases under adding additional unique

solutions, that is, when a solution is added that is not a duplicate of a solution already present in the set;

2. *Twinning*: Diversity is constant under adding a solution that is a duplicate of a solution already present in the set;

3. *Monotonicity in distance*: Diversity increases under more dissimilarity between pairs of solutions, that is, for equal-size sets S, S', it holds that

$$\mathrm{D}(S') \geq \mathrm{D}(S) \iff \mathrm{d}(\mathbf{x}'_i, \mathbf{x}'_j) \geq \mathrm{d}(\mathbf{x}_i, \mathbf{x}_j),\ 1 \leq i, j \leq |S|. \tag{3.5}$$

For obtaining a set diversity measure $\mathrm{D}(S)$ with the properties listed above, [**Solow and Polasky, 1994**] start from the domain-specific distances between each pair of solutions $\mathbf{x}_i, \mathbf{x}_j \in S$ and transform these into similarity scores between 0 and 1,

$$\mathrm{sim}_{\mathrm{d},\theta}(i, j, S) = \exp(-\theta \cdot \mathrm{d}(\mathbf{x}_i, \mathbf{x}_j)),\ \theta > 0. \tag{3.6}$$

Here, θ scales the "reverse" non-linear transformation, a smaller value for θ results in higher similarity scores with less spread. The similarity scores are put in a $|S|$-by-$|S|$ matrix $\boldsymbol{A}$,

$$a_{ij} = \mathrm{sim}_{\mathrm{d},\theta}(i, j, S),\ 1 \leq i, j \leq |S|. \tag{3.7}$$

The set diversity measure termed *Solow–Polasky* [**Ulrich and Thiele, 2011**] then follows by taking the sum of the elements of the inverse of $\boldsymbol{A}$,

$$\mathrm{D}_{\mathrm{SP}}(S) = \sum_{i,j} b_{ij},\ \boldsymbol{B} = \boldsymbol{A}^{-1}. \tag{3.8}$$

$\mathrm{D}_{\mathrm{SP}}(S)$ results in a real number in the interval $[1, |S|]$: When $\mathrm{sim}_{\mathrm{d},\theta}(i, j, S) = 0$ for all $i \neq j$ (all solutions are completely dissimilar), then $\boldsymbol{B}$ (as well as $\boldsymbol{A}$) is the identity matrix and $\mathrm{D}_{\mathrm{SP}}(S)$ is equal to $|S|$; if $\mathrm{sim}_{\mathrm{d},\theta}(i, j, S)$ approaches 1 for all i, j (all solutions are almost perfectly identical), then $\mathrm{D}_{\mathrm{SP}}(S)$ also approaches 1 [**Solow and Polasky, 1994**].

Pareto Front Interpretation

For each solution we thus get a pair of values: Its quality value and the diversity score of the level-set approximation based on its quality value. We illustrate a method's performance from these pairs of values by interpreting them as a *Pareto front* approximation in quality/diversity space, see Figure 3.8.

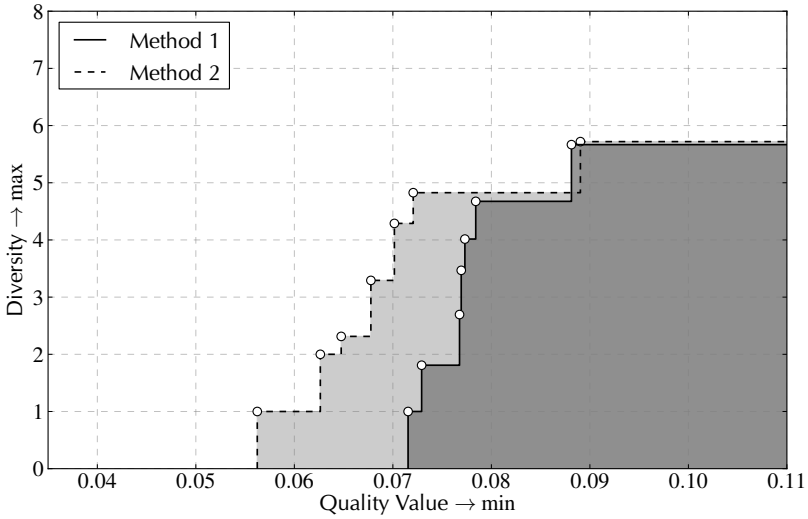


Figure 3.8 Performance Comparison. Per method, of each possible level set in its result set, the value-pair of the quality threshold and diversity score is plotted, governed by the quality values of the solutions in the result set. Interpreting these pairs jointly as a Pareto-front approximation, the dominated surface in quality/diversity space is visualized. Using a reference point of $(0.11, 0)$, method 1 has an obtained hypervolume of 0.184 and method 2 of 0.235.

Allowing for easier comparison, in a further step, the Pareto front is abstracted into a singular value expressing the dominated surface or *hypervolume* [**Zitzler and Thiele, 1998**] in the quality/diversity space that it covers with respect to a *reference point*. The reference point should represent worse than possible quality and diversity scores to make sure that all possible quality/diversity pairs contribute to the hypervolume.

3.3 · Efficiently Finding Diverse High-quality Solutions

Starting from the population-based optimization paradigm of Evolutionary Algorithms (see Chapter 2), the aim is to define a search approach that is able to efficiently produce diverse high-quality solutions. Taking inspiration from the discussion on innovativeness, we envision using an *exploration criterion*, next to the quality criterion, based on *online* novelty or interestingness to help steer the search, see Figure 3.9 and Figure 3.10. The idea is to make the search for diverse solutions more efficient by directing it into novel or interesting areas, and away from areas that are "known" already, based on a memory of *earlier sampled solutions* in the optimization run. Recently, the approach of attaining diversity by rewarding online novelty was also described in [**Lehman et al., 2013**].

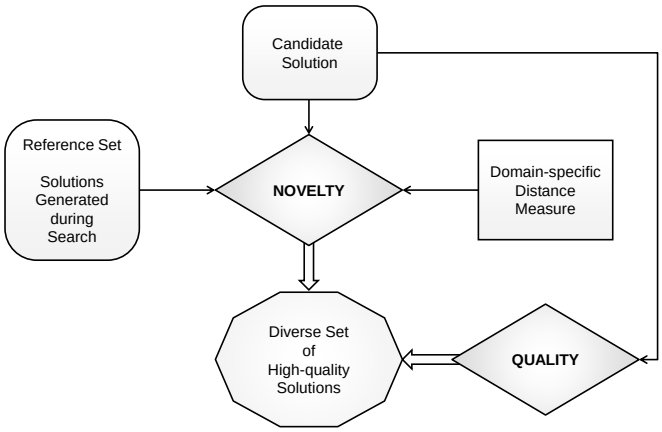


Figure 3.9 Using Novelty for Finding Diverse Solutions. A candidate solution is compared to solutions generated earlier during the search, using a domain-specific distance measure. The resulting novelty value serves as additional criterion in quality-based search, intended for improving exploration in order to assist in finding a diverse set of high-quality solutions.

Importantly, we are re-using ideas that are used for expressing innovativeness, but applied in this way, they do not provide the actual innovativeness of solutions found. In the online setup, the comparison is done to solutions found earlier in the search only, whereas actual innovativeness is derived from the reference set of all state-of-the-art solutions in the application domain.

It is straightforward to see how stepwise selection for novelty promotes domain-specific distance between found solutions, and thus the diversity of the entire set of found solutions with respect to the same domain-specific distance measure. Selecting on interestingness, on the other hand, induces a pattern of exploration that is optimal for obtaining an accurate model of the search space by promoting the learning progress that candidate solutions give rise to. The question is whether this exploration pattern implicitly helps in finding solutions that are diverse with respect to a distance measure that is unrelated to the model, as compared to selecting on novelty that *is* related to the distance measure. Presumably, selecting on interestingness makes the exploration more efficient by moving quickly through areas where the information density is low, thereby helping to find different high-quality solutions.

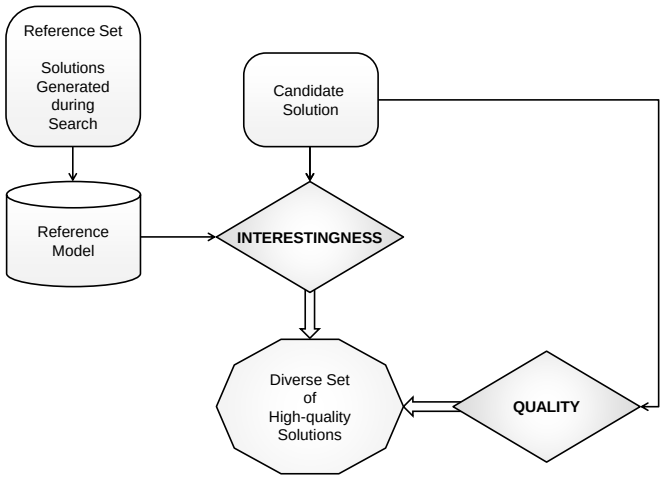


Figure 3.10 Using Interestingness for Finding Diverse Solutions. A reference model is derived from solutions generated earlier in the search. Each candidate solution gets an interestingness score based on the improvement of the model that it leads to. Using interestingness as additional criterion should move the search into areas of the search space with higher information density, thereby implicitly assisting quality-based search in finding a diverse set of high-quality solutions.

3.4 · Summary

We define *innovative solutions* as being *novel* and of tolerable quality, both with respect to the *reference set* of all *state-of-the-art* solutions in the application domain. Novelty is the difference to the closest solution from this set with respect to a *domain-specific distance measure*.

As evaluating novelty and innovativeness requires composing the comprehensive set of state-of-the-art solutions, we instead adopt finding a diverse set of high-quality solutions as search objective. A set of high-quality solutions, different from each other with respect to the domain-specific distance measure mentioned above, potentially contains innovative solutions.

The likeliness of an innovative solution actually being adopted by a human engineer is formalized as its *interestingness*. Interestingness is related to the *learning progress* that a solution leads to with respect to the domain model that exists in the mind of the human domain expert. As in learning, a solution should be sufficiently novel but not too extreme for it to give rise to maximum learning progress. Nevertheless, as interestingness is a highly subjective notion, the aim of the search remains finding innovative solutions (thus with maximum novelty), through optimizing on

diversity and quality.

Lastly, deriving from the discussion on innovativeness, the idea is presented of using novelty and interestingness within the search to efficiently find diverse high-quality solutions. Instead of the set of state-of-the-art solutions, a *dynamic* reference set of solutions generated earlier during the search is used to obtain *exploration criteria* that can be used to steer the search into novel or interesting areas of the search space.

The formulation of such novelty and interestingness-based exploration criteria is worked out in Chapter 4, and in Chapter 5, the appropriate introduction of an exploration criterion into quality-based search is addressed. Chapter 6 describes the application of the developed methods to an airfoil optimization task, including the definition of a domain-specific distance measure.

4

Criteria for Guiding Exploration

For the realm of real-valued design spaces, this chapter defines criteria for design-space exploration, to be used in *iterated adaptive search* methods. By iterated adaptive search we mean that after each iteration in the search, the search distribution from which solutions are sampled is adapted to better meet the search objective. Adaptive search potentially allows for quickly moving through regions that do not require closer investigation as they do not contain a lot of information, provided that it is guided by an *exploration criterion* that properly reflects this. Furthermore, adaptive search has more flexibility in setting the initial search distribution than an approach with a static search distribution, as it is able to self-adapt to appropriate settings. Using adaptive search, however, can cause unintended adaptation behavior in applying a *dynamic* exploration criterion: If the search distribution starts decreasing in magnitude, a scenario is possible where it keeps doing so, as, at these lower scales, the search landscape keeps changing as well, effectively causing exploration to stagnate.

We aim for design-space exploration to occur in a sequential, path-based fashion, in which exploration draws the search to a certain location and then repels it again, away from all earlier visited areas. This is a requirement arising from certain optimization problems in which the search needs to be started from a single, fixed solution (see Chapter 6). Right after the attraction phase, before being repelled, the search can turn to exploitation of the quality function. To be drawn to a certain location and then repelled from it, the exploration criterion needs to keep track of the *history* of where the search has been before, and is thereby inherently dynamic. Exploration needs a global view of the search space, whereas exploitation uses a *greedy*, local view, just comparing to the previous state.

This chapter provides an overview of dynamic exploration measures that are

based on ideas of *novelty* of solutions with respect to the history of visited solutions. Different views on novelty are presented, based, roughly, on a *distance measure* and on fitting a *surrogate model*. The model-based approach can be further used to derive *interestingness* of solutions, a more concise expression of the amount of useful knowledge that a solution introduces in the model.

In novelty based on a distance measure, a comparison is made to points in a *reference set* of solutions, whereas in novelty based on the prediction of a model, the surrogate model is generated from the reference set. It is the reference set that makes the novelty expression dynamic: During exploration, the reference set is updated with new solutions that were encountered. A static formulation of both variants of novelty *is* possible though with respect to an *external* reference set of solutions that were not generated while exploring. In using novelty as exploration criterion, one has the choice between

- comparing to a dynamic set of solutions generated earlier during the run;
- comparing to a static set of a number of known established solutions, to which the difference is to be maximized;
- comparing to a single solution only, from which maximally-different solutions are to be generated.

In moving away from a single, e.g., patented solution, incremental steps (i.e., successive exploration phases) may be necessary for the search to successfully traverse the search space, advocating the use of a dynamic set of sampled solutions instead of a single established solution as reference set: In using a single solution as reference, the necessary stepping stones for putting distance to this to-be-avoided solution may be obscured, as a sequence of seemingly counter-productive steps can be required. We therefore limit ourselves to the dynamic formulation of the exploration measures, derived from solutions generated during the search.

From Action-space Exploration to Design-space Exploration

Typically, in developmental robotics, large state spaces are to be searched efficiently through *reinforcement learning*. Whereas *supervised learning* is a passive approach in which an agent does not have influence on the data presented to it for training, in reinforcement learning, the agent itself actively chooses the data for which it

wants to receive feedback to build its understanding of the environment [**Sutton and Barto, 1998**, **Barto and Dietterich, 2004**]. Reinforcement learning generally tries to optimize an agent's explorative behavior in its environment [**Rückstieß et al., 2010**]. The assumption underlying this *active learning* is that if the learning algorithm is allowed to choose the data from which it learns itself, it can achieve high accuracy using as few training instances as possible [**Settles, 2010**].

The idea of using novelty and interestingness of possible actions or solutions as an indication of what to try or where to explore next is widespread in *neuro-control* and *reinforcement learning* [**Schmidhuber, 1991**, **Schmidhuber, 1997**, **Cuccu et al., 2011b**, **Luciw et al., 2011**, **Graziano et al., 2011**, **Oudeyer et al., 2007**, **Oudeyer and Kaplan, 2007**, **Baranes and Oudeyer, 2013**, **Hester and Stone, 2012**, **Lehman and Stanley, 2011**]. To a lesser extent, it is being proposed for design-space exploration in *black-box optimization* [**Graening et al., 2010**, **Schaul et al., 2011**, **Cuccu et al., 2011a**]. A division is made between similarity-derived, *distance-based* novelty, and *learning-based* novelty related to fitting a surrogate model (in [**Graziano et al., 2011**] referred to by *coherence* and *compression*, respectively).

Learning-based novelty infers information from a reference set of performed actions through generating an internal model. Situations, based on actions taken, that could not have been predicted by the agent from previous experience are defined as novel, expanding its understanding of the environment. Situations that could have been predicted from previous experiences, despite never actually having been experienced, are not novel under this definition [**Gero and Saunders, 2000**]. The agent should choose actions for which the expectation of improvement of the model is high [**Schmidhuber, 1991**]. This interestingness of an action is based on its novelty, but is also related to how well an agent can adopt the information gained from novel experiences. New experiences similar but different from what was experienced before provide the best opportunity for rapid learning [**Saunders and Gero, 2004**]. Interestingness is the steepness of the learning curve [**Schmidhuber, 2009**], and by selecting on it we can maximize the *knowledge gain*.

This chapter provides an overview of novelty and interestingness expressions from the perspective of design-space exploration, by unifying the expressions in notation and proposing adjustments with this application in mind where necessary.

Interestingness in Data Mining

Interestingness measures in *data mining* are based on the same underlying ideas as discussed above. These measures express interestingness of found *association rules* to determine the most-relevant rules [**Hébert and Crémilleux, 2007**]. Several pattern discovery methods produce many obvious or irrelevant patterns, and do not fully account for valuable, prior domain knowledge that decision makers have [**Padmanabhan and Tuzhilin, 1998**]. For dealing with this, there are *objective* interestingness measures that depend on the structure of the found pattern and underlying data, and *subjective* interestingness measures that compare properties of a pattern with user beliefs or biases regarding relationships in the data [**Silberschatz and Tuzhilin, 1996, Saunders, 2002, Hilderman and Hamilton, 2003**]. An overview of objective and subjective interestingness measures in data mining is given in [**Geng and Hamilton, 2006**].

4.1 · Exploration Measures in Adaptive Search

In the following sections, we present different measures for inducing sequential exploration within an adaptive search method. On a two-dimensional artificial test function, visual analysis is performed of the sampling dynamics of selecting solutions using the exploration measure only: In the selection, no quality criterion is involved. Promising measures with respect to sampling dynamics, which is further specified below, will be selected for performance assessment when integrated as additional criterion into quality-based search in Chapter 5.

Test Setup

As test function a two-dimensional instance of a composite function termed *SineUnlearnable* is used, see Figure 4.1, with search domain $[-15,15]^n \subseteq \mathbb{R}^n$, $n = 2$,

$$f_{\text{SineUnlearnable}}(\mathbf{x}) = \begin{cases} \mathcal{U}(-5,5) & \text{if } -2 < x_i < 2 \text{ for } 1 \leq i \leq n \\ -\frac{1}{n}\sum_{i=1}^{n} \sin(x_i) & \text{otherwise} \end{cases} . \quad (4.1)$$

Test function SineUnlearnable has a cross-shaped region with extreme *noise* (i.e., uncertainty with an intrinsic stochastic nature [**Kruisselbrink, 2012**]) within which $f_{\text{SineUnlearnable}}$ takes on a different random value between -5 and 5 for each evaluation

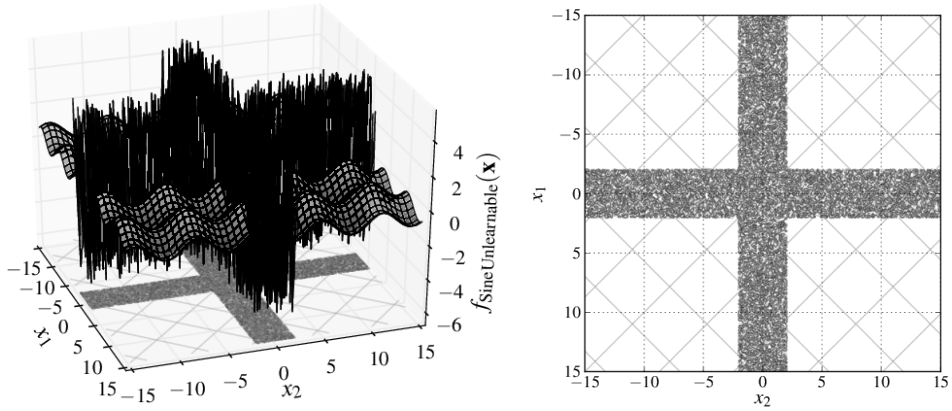


Figure 4.1 Test Function *SineUnlearnable*. Composite test function that consists of a multivariate sine function on the one hand and a cross-shaped area with extreme noise (i.e., function values in this region differ for each evaluation of the test function) on the other hand.

of $f_{\text{SineUnlearnable}}$. In the rest of the search domain, the composite function behaves as a multivariate sine function.

The search is initialized in point $(0,0)$ in the middle of the *unlearnable* area, meaning that the region features too excessive noise to be concisely approximated using a surrogate model. Some of the exploration measures that are presented rely on approximation models (see Chapter 2), and we initialize in this region to determine whether these measures are able to make the search escape from it. When integrated in quality-based search, the exploration measures are to assist in finding a diverse set of high-quality solutions. We are thus interested in the pattern of exploration that is induced after the noisy region has been left, which will be judged by the ability of leading to diverse solutions.

As underlying search method, a $(3,6)$-CMA-ES is used with comma-selection and populations of 3 parents and 6 offspring (population sizes prescribed for two-dimensional search spaces, see Chapter 2), and an initial stepsize of 0.3. For constraint handling, we add a variable penalty to the criterion that is selected on, here the exploration measure, that increases linearly with the extent of violation of the lower boundary of -15 or the upper boundary of 15 per dimension.

For the exploration measures that employ approximation models, the used modeling technique and accompanying settings are described in the section defining the measure. All methods and techniques used have been implemented using the

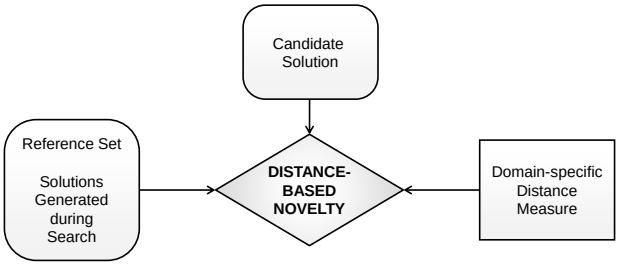


Figure 4.2 Distance-based Novelty. A candidate solution is scored on novelty, based on its distance to the closest solutions in an archive of earlier encountered solutions during the search. A domain-specific distance measure that reflects the distinctiveness of solutions is required for the comparison.

Shark Machine Learning Library v2.3.43[1].

The remainder of this chapter is organized as follows: Section 4.2 presents exploration measures that directly express novelty of solutions, which will be categorized in distance-based and learning-based novelty. Section 4.3 then describes exploration measures that process learning-based novelty values further to express the interestingness of solutions. To be able to concisely describe and compare the different interestingness measures, a notational scheme is given as well as a means of determining a *regional error value* from observed learning-based novelty values. In Section 4.4, a summary of the chapter is given, presenting two promising exploration measures for integration in quality-based search.

4.2 · Novelty

To determine a candidate solution's distinctiveness or novelty from the knowledge gathered on the design space, an appropriate representation of the available knowledge is required such that new solutions can be compared to it. Possible representations of the available knowledge are an *archive* containing all sampled solutions, and a *surrogate model* that learns a mapping between different characteristics of a solution, for instance from the solution vector in the search space to the quality value that becomes available after evaluation.

Novelty as it is introduced in Chapter 3 uses the distance to solutions in a *reference set* with respect to a *domain-specific distance measure* as novelty value. We will refer

[1] http://shark-project.sourceforge.net

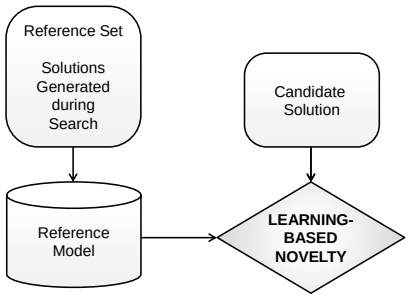


Figure 4.3 Learning-based Novelty. Based on a reference set of solutions encountered during the search, a reference model is generated that approximates the mapping between solution vectors and quality values. For a candidate solution, the error that the reference model makes in predicting its quality value serves as novelty value.

to this first type of novelty as *distance-based* novelty, see Figure 4.2, using an archive of sampled solutions as reference set. Alternatively, using the archive as reference set, one generates an approximation model and the error that the model makes in its approximation for a certain solution serves as its novelty value. We will refer to this type of novelty as *learning-based* novelty, see Figure 4.3. Chapter 3 associates the usage of a reference model with calculating interestingness but more accurately, in determining interestingness, the learning-based novelty based on two reference models is compared, see Section 4.3.

Selecting on distance-based novelty stepwise promotes domain-specific distance between solutions found and thus the diversity of the entire set of solutions found, with respect to the same domain-specific distance measure. In contrast, in selecting on learning-based novelty, the idea is that solutions with maximum error improve the surrogate model most after including them in training, maximizing *learning progress*. A scheme aimed at maximizing learning progress realizes this through estimating a pattern of exploration that is optimal for efficiently obtaining an accurate model of the search space. Potentially, in doing so, it implicitly samples solutions that are distant from each other with respect to the domain-specific distance measure, thereby constituting a diverse set.

4.2.1 · Distance-based Novelty

A straightforward way of representing knowledge gathered on the search space is through an archive of the solutions sampled during the search process. New solutions get a novelty score based on their distance to the closest point(s) in this archive. This

requires, however, an appropriate measure to indicate the distance between solutions. This work deals with search spaces consisting of real-valued vectors, therefore a generally-applicable distance formulation is the Euclidean distance between the solution vectors.

Certain optimization problems involve additional representations of solutions that better reflect their distinctiveness. Consider for instance a controller that is to steer a robot in navigating a maze, described in [**Lehman and Stanley, 2008**]. Here, the solution vector is a parameterization of the *artificial neural network* that controls the robot. However, the controller can be argued to be better represented by the end position in the maze that it gives rise to. Hence, [**Lehman and Stanley, 2008**] adopt the Euclidean distance between end positions as distance measure, instead of the Euclidean distance between the solution vectors. Thus, to come to a domain-specific distance measure, choose a characterizing representation first, and then a way of calculating the distance between two instances of that representation.

Distance-based novelty was put forward by [**Lehman and Stanley, 2008**] as a *replacement* objective to drive optimization, thereby ignoring the quality of solutions. There are no pre-defined fitness functions in nature; rather, there is ad-hoc fitness determination that is subject to change during the evolutionary process. Therefore, to artificially reproduce *open-ended evolution* that leads to increasing complexity in generated solutions, one can consider abandoning the a-priori quality objective and search for novel solutions instead. Not all optimization problems are suitable, however, for this pure novelty-search approach [**Cuccu and Gomez, 2011**]: Novelty search becomes computationally feasible in a scenario in which an appropriate distance measure reduces the effective size of the search space [**Lehman and Stanley, 2008**].

Alternatively, novelty and quality are jointly optimized on, which was shown to produce better results than pure novelty search in [**Mouret, 2011, Cuccu and Gomez, 2011, Lehman et al., 2013**]. In our used test setup, the optimization is on exploration measure only like in pure novelty search, but, as stated before, after the initial analysis the most-promising exploration criteria will be integrated into quality-based search in Chapter 5.

Interestingly, [**Hester and Stone, 2012**] employ another variant of pure novelty search: They optimize on distance-based novelty and learning-based novelty in parallel. A model is generated by using a learning-based novelty expression (similar to *dispersion in predictions*, see Section 4.2.2.3) that favors solutions with high modeling

error. This allows for efficient modeling, as the solutions for which the model can be improved greatly are selected. On the other hand, a distance-based novelty expression (*uniqueness*, see Section 4.2.1.2) is used to drive the exploration towards unvisited areas, to improve the global covering of the search space by the model.

4.2.1.1 · Sparseness

In [**Lehman and Stanley, 2008**], *sparseness* (Sp) is proposed as distance-based novelty measure. A candidate solution $\mathbf{x}$ is compared to solutions $\mathbf{q}$ from the union Q of a partial archive S of earlier sampled solutions *and* the set $X \setminus \{\mathbf{x}\}$ of all current candidate solutions excluding $\mathbf{x}$.

Sampling a solution means evaluating it on quality, and the archive S consists of tuples of solutions with accompanying quality value $(\mathbf{q}, f(\mathbf{q}))$. We use this definition of sampling for the sake of generality, although distance-based novelty does not inherently require evaluating solutions on quality, this depends on the domain-specific distance measure that is used.

The closest k neighbors of $\mathbf{x}$ are determined based on a domain-specific distance measure $\mathrm{d}(\mathbf{x}, \mathbf{q})$,

$$\underset{\{\mathbf{q}_1,\ldots,\mathbf{q}_k\} \subseteq Q}{\operatorname{argmin}} \sum_{i=1}^{k} \mathrm{d}(\mathbf{x}, \mathbf{q}_i), \tag{4.2}$$

$$Q = \{\mathbf{q} \mid (\mathbf{q}, f(\mathbf{q})) \in S\} \cup X \setminus \{\mathbf{x}\}. \tag{4.3}$$

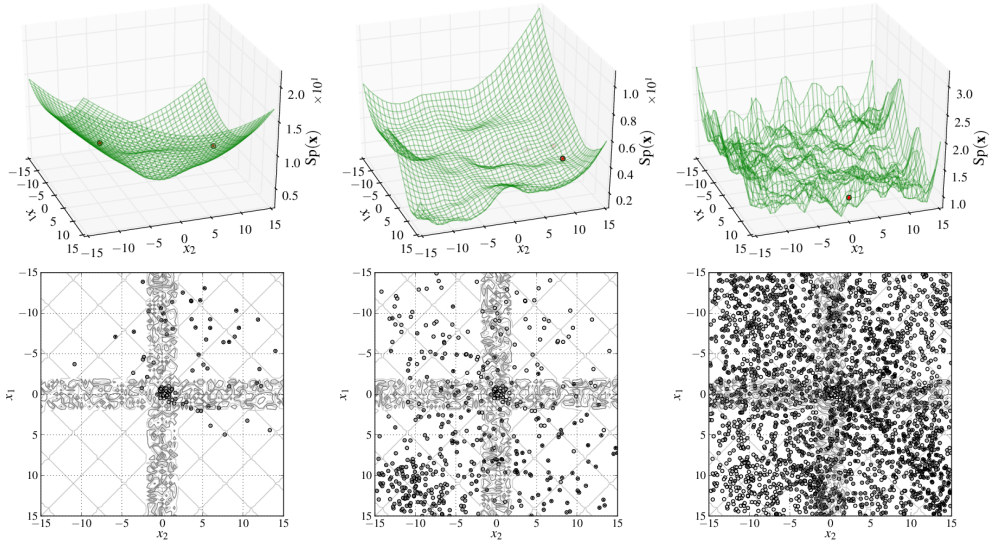
The sparseness of $\mathbf{x}$ is then taken as the average distance to these closest k solutions,

$$\mathrm{Sp}(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^{k} \mathrm{d}(\mathbf{x}, \mathbf{q}_i). \tag{4.4}$$

Settings for the following parameters are required, the listed settings are from [**Lehman and Stanley, 2008**, **Lehman and Stanley, 2011**]:

- Number of closest solutions k: A value of 15 is used by Lehman and Stanley for varying problems;

- Dynamic archive-inclusion threshold $\mathrm{Sp}_{\min}$: Only solutions with sparseness that exceeds this threshold are added to the archive. After some initial setting (we use 1.0), $\mathrm{Sp}_{\min}$ is lowered by 5% if $\frac{1}{100}$-part of the sampling budget was used



(a) Generation 20 **(b)** Generation 100 **(c)** Generation 495

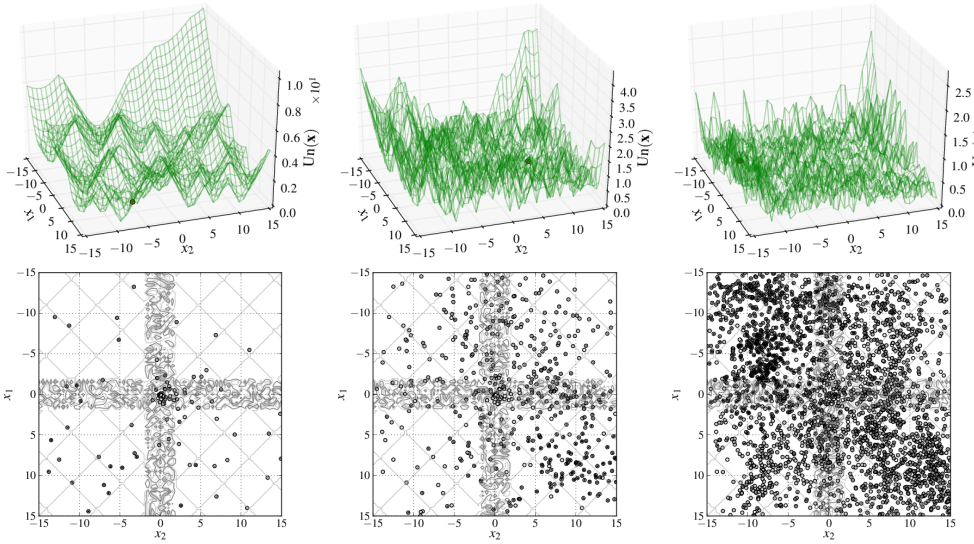
Figure 4.4 Sparseness. The bottom-row plots show the solutions generated by a single run on test problem SineUnlearnable, of a (3,6)-CMA-ES selecting on sparseness, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored). The top-row plots display the development of the sparseness scores, where the red dots represent the lastly-selected parent population. Selection on sparseness stepwise generates an evenly-spaced sampling that gets finer-grained as the run continues.

and no solutions were added; it is increased by 20% if $\frac{1}{100}$-part of the sampling budget was used and more than 4 solutions were added;

- Each sampled solution has an unconditional 0.1% chance to be added to the archive.

In Figure 4.4, the solutions generated in selecting on sparseness are displayed as well as the development of the sparseness scores, using the Euclidean distance between solution vectors as distance measure. In using this distance measure, the sampling behavior is unrelated to the quality function and leads to an evenly-spaced covering that is generated in segments, and that becomes finer-grained as selection on sparseness is continued.

Employing sparseness requires settings for a number of parameters, most of them involved in the update scheme for the dynamic archive-inclusion threshold. Lehman and Stanley do not provide clear guidelines on initializing the parameters, but simply



(a) Generation 20 **(b)** Generation 100 **(c)** Generation 495

Figure 4.5 Uniqueness. The bottom-row plots show the solutions generated by a single run on test problem SineUnlearnable, of a (3,6)-CMA-ES selecting on uniqueness, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored). The top-row plots display the development of the uniqueness scores, where the red dots represent the lastly-selected parent population. Selecting on uniqueness, like selecting on sparseness, stepwise generates an evenly-spaced sampling that gets finer-grained as the run continues.

re-using their reported settings results in the aimed-for ongoing oscillating sampling behavior that generates diverse solutions.

4.2.1.2 · Uniqueness

Seeking to reduce the complexity of the scheme and the number of parameters involved, we reformulate sparseness as the *uniqueness* (Un) of a solution. The uniqueness is taken as the minimum distance that occurs to a solution $\mathbf{q}$ in an archive S containing *all* sampled solutions,

$$\mathrm{Un}(\mathbf{x}) = \min_{(\mathbf{q}, f(\mathbf{q}))\in S} \mathrm{d}(\mathbf{x}, \mathbf{q}). \tag{4.5}$$

This definition of distance-based novelty is also used in [**Hester and Stone, 2012**, **Hester, 2012**].

In Figure 4.5, the solutions generated in selecting on uniqueness and the development of the uniqueness scores are displayed, using the Euclidean distance between

solution vectors as distance measure. Like in selecting on sparseness, the sampling behavior leads to an evenly-spaced covering that gets finer-grained sequentially.

In expressing uniqueness, there is no difference in novelty score in case the *closest solution* $\mathbf{q}$ has no similar counterparts in the archive, compared to when the closest solution is part of a group of similar archived solutions. However, as all sampled solutions are added to the archive, this effect is of less consequence than it would be in a sparseness scheme with $k = 1$, as the chance for a solution of being included in the archive there depends heavily on the moment during the search that it is generated. Furthermore, the unconditional archive leads to a higher computational load than the partial archive in sparseness, but the archive size is of course bounded by the number of allowed to-be-sampled solutions (i.e., the quality-function evaluation budget).

4.2.2 · Learning-based Novelty

Maximizing distance-based novelty leads to diverse solutions with respect to the same distance measure, when this is the Euclidean distance between solution vectors. Selection on distance-based novelty is then likely to move through the search space at an optimal speed when applied in methods with adaptive search distributions, as simply the magnitude of the search distribution is increased. For a different domain-specific distance measure that for instance is not determined over the solution vectors, this is not necessarily the case. In such a scenario, we could possibly move through the search space more efficiently in maximizing learning progress.

The naive approach of maximizing learning progress is through maximizing learning-based novelty [**Thrun, 1995**]. Learning-based novelty requires no explicit distance measure: The distinctiveness from available knowledge is represented as the *generalization performance* that a surrogate model trained on the available knowledge has in approximating a to-be-learned mapping for an unseen solution. The error that the surrogate model makes in its prediction for a candidate solution serves as novelty expression [**Thrun, 1995**, **Oudeyer and Kaplan, 2007**, **Graening et al., 2010**, **Cuccu et al., 2011b**]. Typically, we will approximate the mapping from solution vectors to quality values. By including a solution in model training for which the predicted quality value showed high discrepancy with the actual quality value, the assumption is that we are able to greatly improve the model, thereby realizing high learning progress. Question is whether this approach is sufficient for implicitly finding diverse solutions, or that a more elaborate approach of estimating learning progress is required.

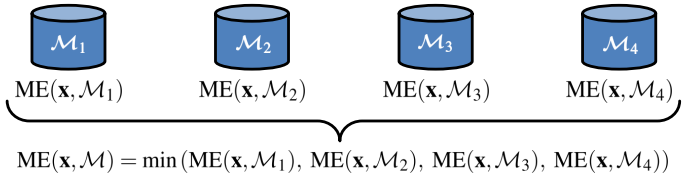


Figure 4.6 Ensemble of Local Models. A global model is emulated using an ensemble of local models. Each local model uses data from a different area in the search space for training. The global modeling error $\mathrm{ME}(\mathbf{x},\mathcal{M})$ is obtained by evaluating on all local models and adopting the lowest modeling error.

Ensemble of Local Models

To be prepared for modeling irregular quality landscapes, we extend the learning-based novelty approach with a *global model* $\mathcal{M}$ emulated through an *ensemble of local models* $\mathcal{M}_i$ [**Reehuis et al., 2011**], see Figure 4.6. A global model approximates the entire search space, whereas a local model is aimed at only accurately approximating a certain part of the search space. The different local models are trained on disjoint or partly shared data. The global modeling error for a candidate solution $\mathbf{x}$ is obtained by evaluating it on all local models and adopting the *lowest* resulting modeling error,

$$\mathrm{ME}(\mathbf{x},\mathcal{M}) = \min_i \mathrm{ME}(\mathbf{x},\mathcal{M}_i), \tag{4.6}$$

originating from the local model that provides the most-accurate approximation for $\mathbf{x}$.

Each generation, we train a new local model on the last γ generations of sampled solutions. Every γ-th local model is permanently stored. Together with these stored models, the last local model generated is used within the ensemble. The question remains how many training points to use for generating the local models. A sufficient number of points is required and these should not be too distant from each other. In [**Loshchilov et al., 2012**], some guidelines are provided depending on the dimensionality of the input space, the used test function, and the used modeling technique. We use $\gamma = 5$ generations of sample data for training, which in using the $(3,6)$-CMA-ES corresponds to maximally 30 training points per local model. This is deemed sufficient for generating local models of the two-dimensional test function. On a higher-dimensional problem, one can use a lower value of γ to make sure that the points are not spread out too far. This is possible when in higher dimensions, more solutions are sampled per generation: For the CMA-ES, larger population sizes are prescribed in higher dimensions (see Chapter 2). Only solutions that lie within the lower and upper boundaries, set for the search space, are included in training. Note

that as we are using an adaptive search method, the distance between training points and thereby the locality of the derived model may show high variance between local models.

The proposed *horizontal* ensemble is different from the *vertical* model ensembles discussed in, e.g., [**Jin and Sendhoff, 2004**] and [**Lim et al., 2010**]. There, an average prediction is calculated using multiple vertical (local) models, trained on the same data, in order to increase prediction fidelity. We do use a similar approach of a vertical stack of local models in one of the learning-based novelty expressions, see Section 4.2.2.3.

4.2.2.1 · Prediction Error

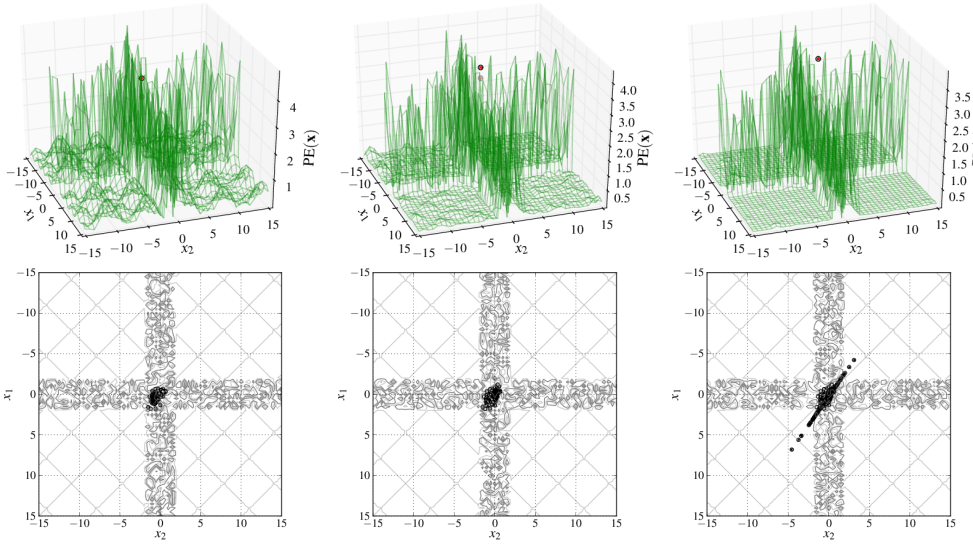
For a surrogate model that approximates a mapping from solution vectors $\mathbf{x}$ to quality-function values $f(\mathbf{x})$, the *prediction error* (PE) is calculated as the absolute difference between the model's predicted quality value $\hat{f}(\mathbf{x})$ and the actual outcome $f(\mathbf{x})$ of evaluating $\mathbf{x}$ by the quality function [**Oudeyer and Kaplan, 2007**, **Graening et al., 2010**],

$$\mathrm{PE}(\mathbf{x}) = \left| \hat{f}(\mathbf{x}) - f(\mathbf{x}) \right| . \tag{4.7}$$

Given a model $\mathcal{M}$, $\mathrm{PE}(\mathbf{x})$ is the exact formulation of the modeling error $\mathrm{ME}(\mathbf{x}, \mathcal{M})$.

Within the horizontal ensemble of local models, we employ *feed-forward neural networks* (FFNNs, see Chapter 2) as local modeling technique. The FFNNs have linear outputs and a hidden layer of 10 sigmoidally-activated nodes, a bias node, and are trained using improved Rprop [**Igel and Hüsken, 2003**] for a maximum of 10 epochs from random initial connection weights in $[-0.1, 0.1]$. They are fully connected, including direct connections from input to output nodes. All data is used for training, no validation sets are used. The local models should jointly represent the current knowledge of the quality landscape best, hence there is no risk of *overfitting*.

In Figure 4.7, the solutions generated in selecting on PE are displayed, as well as the development of the PE scores. Most notably, selecting on PE the search is not able to leave the unlearnable area featuring extreme noise. The prediction error remains constantly high despite of sampling, but this is in line with the definition and calls for a mechanism to cope with areas that cannot be modeled, which PE does not include. Furthermore, an unintended artifact of using the horizontal ensemble can be observed: As the generations pass, the prediction error decreases in areas where no points have been sampled. As more local models are added to the ensemble, the global modeling



(a) Generation 20 **(b)** Generation 100 **(c)** Generation 495

Figure 4.7 Prediction Error. The bottom-row plots show the solutions generated by a single run on test problem SineUnlearnable, of a (3,6)-CMA-ES selecting on PE, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored). The top-row plots display the development of the PE scores, where the red dots represent the lastly-selected parent population. In selecting on PE, the search is unable to leave the area with extreme noise.

error is chosen out of more predictions, and thus a better approximation of a candidate solution's quality value becomes more likely.

Estimating the Prediction Error

Ranking on PE requires evaluating all candidate solutions on quality. Ideally, one only evaluates the most-promising candidates. Therefore, [**Thrun, 1995**] and [**Oudeyer et al., 2007**] describe an approach of using an additional *error model*, next to the quality model, that estimates the prediction error. Instead of on PE, solutions are ranked on the *estimated prediction error* $\widehat{\mathrm{PE}}(\mathbf{x})$ and only selected solutions get evaluated on quality. For these sampled solutions, the actual prediction error is then calculated with the sole purpose of training the error model, so it can predict the PE better for the next round of candidates.

In the following subsections, we present other approaches for estimating the prediction error. In these schemes only selected solutions are to be evaluated on quality as well, but they do not require training an additional error model.

4.2.2.2 · Predictive Variance

In using *Gaussian Process modeling* (GP) and *Kriging* approaches to surrogate modeling (see Chapter 2), next to the predicted value $\hat{f}(\mathbf{x})$ (*predictive mean*), an estimate of the associated prediction error is readily available as the *predictive variance* (PV) [**Rasmussen and Williams, 2006, Jones et al., 1998**],

$$\mathrm{PV}(\mathbf{x}) = \sigma^2(\mathbf{x}) \text{ [GP]} \;\hat{=}\; s^2(\mathbf{x}) = \mathrm{MSE}(\mathbf{x}) \text{ [Kriging]}, \tag{4.8}$$

assuming the actual quality value $f(\mathbf{x})$ to be a realization of $\mathcal{N}\left(\hat{f}(\mathbf{x}), \mathrm{PV}(\mathbf{x})\right)$.

PV is presented by [**Schaul et al., 2011**] as expressing interestingness through showing a direct relationship from PV to what they term *expected information gain*, i.e., estimated learning progress,

$$\Psi(\mathbf{x}) = \frac{1}{2}\log \mathrm{PV}(\mathbf{x}) - \frac{1}{2}\log \sigma^2_{\text{noise}}, \tag{4.9}$$

where the noise variance σ^2_{noise} [**Rasmussen and Williams, 2006**] is constant, and thus it holds that $\mathrm{PV}(\mathbf{x}_1) < \mathrm{PV}(\mathbf{x}_2) \iff \Psi(\mathbf{x}_1) < \Psi(\mathbf{x}_2)$ for any two solutions $\mathbf{x}_1, \mathbf{x}_2$. We view PV as an expression of learning-based novelty, as PV, like PE, does not include a mechanism for dealing with regions of the search space that effectively cannot be modeled.

Furthermore, although PV provides an estimate of the modeling error and is thus grouped under learning-based novelty, ranking solutions on PV is governed by the spatial distribution of the sample points only, that is, ranking on PV is actually distance-based. We define *ruggedness* in a certain region of the quality landscape as the variation there in the outcome of the quality function. Although PV depends on the ruggedness of the quality landscape, *relatively*, PV depends only on the distribution of the sample points used in training, see Figure 4.8, because of the assumption that the ruggedness is uniform in each region of the input space (*stationarity assumption*). *Local* predictive variance that takes *local* ruggedness into account could be obtained by using a *non-stationary* variant of GP or Kriging [**Xiong et al., 2007**], or by re-training the surrogate for each candidate solution using its closest neighbors only from the available training data [**Emmerich et al., 2006**].

In our test setup, we follow [**Schaul et al., 2011**] who do not explicitly consider the stationarity assumption and rank multiple solutions using a single model, although we do use multiple local models within the horizontal ensemble (discussed in the

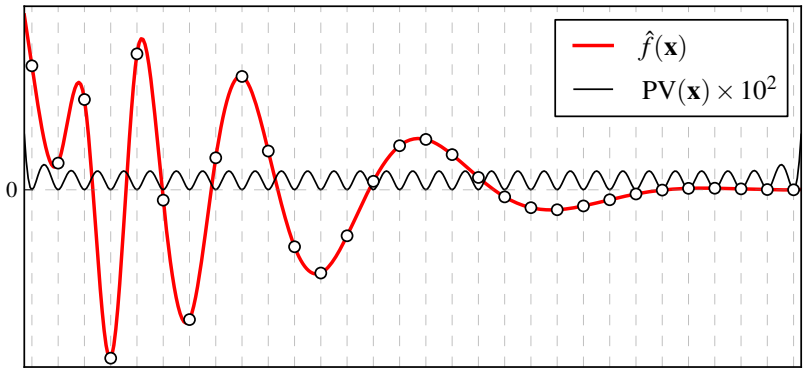


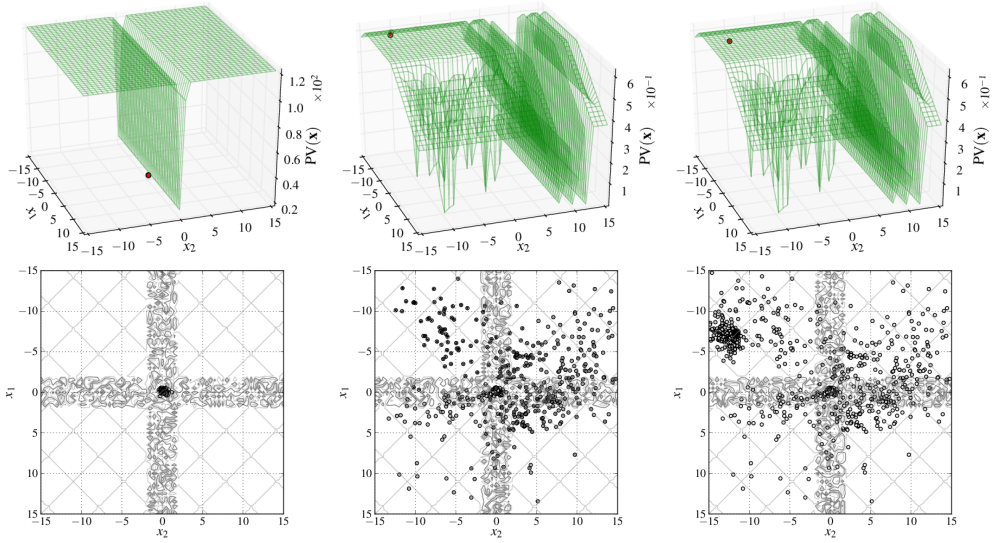
Figure 4.8 Predictive Variance Illustrated. Predictive mean $\hat{f}(\mathbf{x})$ and predictive variance $\mathrm{PV}(\mathbf{x})$ displayed for training points sampled equidistantly from a quality function with varying irregularity. As can be seen from the black line, relatively, predictive variance depends on the spatial distribution of the sample points only.

beginning of Section 4.2.2). There *is* non-stationarity *between* the local models in the horizontal ensemble, but *within* each local model, the stationarity assumption holds. Ranking multiple solutions by taking local irregularity into account, instead of density only, is therefore not possible using the same local model.

Ordinary Kriging (see Chapter 2) is employed[2] with a power-exponential multi-parameter kernel, trained by tuning the θ_i in $[10^{-10}, 10^2]$ and the p_i in $[1,2]$ using *Differential Evolution* [**Storn and Price, 1997**] with population size 50, for 50 generations, followed by a hill-climbing phase. All data is used for training, no validation set is used.

In Figure 4.9, the solutions generated in selecting on PV are displayed, as well as the development of the PV scores. In our density-based setup, the search is able to leave the noisy region in selecting on PV. However, in doing so, it is probably hindered by the horizontal-ensemble approach leading to lower PV values from local models trained on samples adjacent to, but outside of the noisy region, as compared to local models trained on samples from within the noisy region. The non-stationarity between the local models within the ensemble leads to PV scores of different magnitude, clearly visible from generation 100 on. In generation 495, the search appears to be stuck, zooming in on an area of relatively low PV compared to other regions in the search space. This is likely to be due to the interplay of a dynamic exploration measure on the one hand and an adaptive search distribution on the other hand, in which the *search*

[2] The used Kriging implementation was provided by Giles Endicott.



(a) Generation 20 **(b)** Generation 100 **(c)** Generation 495

Figure 4.9 Predictive Variance. The bottom-row plots show the solutions generated by a single run on test problem SineUnlearnable, of a (3,6)-CMA-ES selecting on PV, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored). The top-row plots display the development of the PV scores, where the red dots represent the lastly-selected parent population. While in selecting on PV, the search is able to leave the area with extreme noise, in a later phase it zooms in on an area with relatively low PV with other higher-PV regions around.

horizon, i.e., the solutions reachable by the search in a single step, can decrease too far to move out again.

4.2.2.3 · Dispersion in Predictions

The modeling error $\mathrm{ME}(\mathbf{x}, \mathcal{M})$ of a given model $\mathcal{M}$ in point $\mathbf{x}$ depends on the local variation in the outcome of the quality function around $\mathbf{x}$ and on the spatial distribution of the solutions used in training the surrogate model. As such, we propose to estimate $\mathrm{ME}(\mathbf{x}, \mathcal{M})$ using a *vertical* ensemble of multiple surrogate models $\mathcal{M}_j$, trained on the *same* data [**Jin and Sendhoff, 2004**], and measuring the variation in their predictions for $\mathbf{x}$ [**Reehuis et al., 2011**] (independently proposed in [**Hester and Stone, 2012**, **Hester, 2012**]). In presence of sufficient data, the predictions should largely align, while otherwise be more diverse.

For expressing the variation between the predictions, we determine their *interquartile range* (IQR), which is the difference between the third and first quartile of the

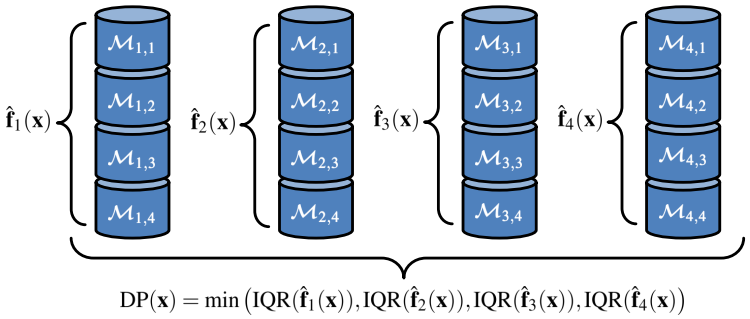


Figure 4.10 Ensemble of Local Model Stacks. A global model stack is obtained using an ensemble of local model stacks. Within local stack i, all surrogate models $\mathcal{M}_{i,j}$ are trained on the same training data, while different local stacks use data from different areas in the search space. The lowest DP score is selected after evaluating a solution $\mathbf{x}$ on all stacks.

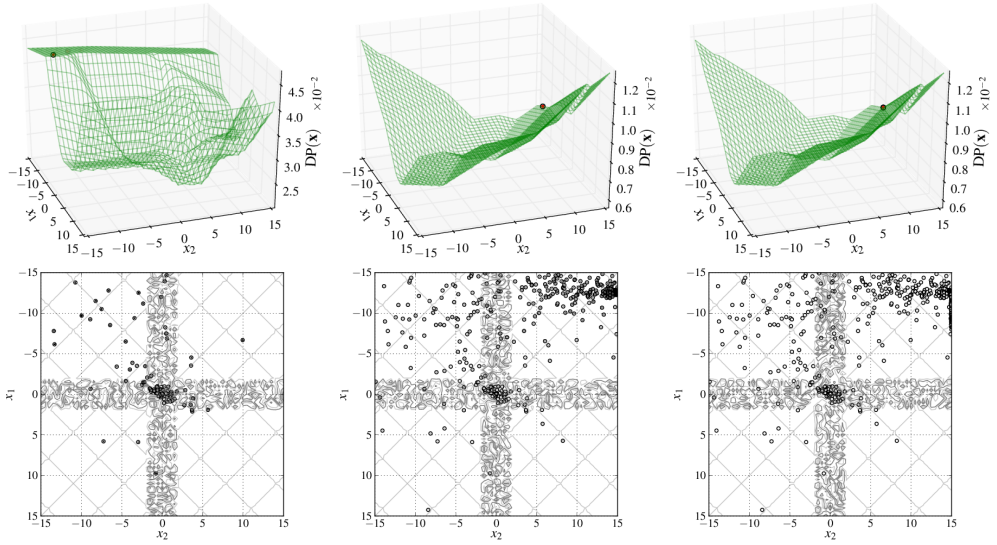
values and is chosen as statistic because it is not prone to outliers. We term this estimation of the modeling error *dispersion in predictions* (DP), calculated over the quality predictions $\hat{f}_j(\mathbf{x})$ by the $\dim\left(\hat{\mathbf{f}}(\mathbf{x})\right)$ models in the vertical ensemble, or model stack [**Reehuis et al., 2011**, **Reehuis et al., 2013a**, **Reehuis et al., 2013c**],

$$\mathrm{DP}(\mathbf{x}) = \mathrm{IQR}\left(\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_{\dim\left(\hat{\mathbf{f}}(\mathbf{x})\right)}(\mathbf{x})\right). \tag{4.10}$$

DP is assumed to provide a similar ordering between candidate solutions as ranking them on their actual PE would, without the requirement of evaluating candidate solutions on quality beforehand; formally, for any two solutions $\mathbf{x}_1, \mathbf{x}_2$ it is assumed that

$$\mathrm{PE}(\mathbf{x}_1) < \mathrm{PE}(\mathbf{x}_2) \iff \mathrm{DP}(\mathbf{x}_1) < \mathrm{DP}(\mathbf{x}_2). \tag{4.11}$$

DP requires surrogate models with potentially differing outcomes, either a combination of different modeling techniques or multiple instances of a technique that may vary in prediction outcome when differently initialized. In [**Jin and Sendhoff, 2004**], neural networks of varying structure are used, whereas [**Hester and Stone, 2012**] use a collection of *decision trees* (i.e., a *random forest*) that are each trained on a different, partly-overlapping subset of the available data. We use 10 stacked instances of the same FFNNs that was used for PE, all sharing the same architecture but initialized from different random weights and with the training points supplied in varying random order.



(a) Generation 20 **(b)** Generation 100 **(c)** Generation 495

Figure 4.11 Dispersion in Predictions. The bottom-row plots show the solutions generated by a single run on test problem SineUnlearnable, of a (3,6)-CMA-ES selecting on DP, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored). The top-row plots display the development of the DP scores, where the red dots represent the lastly-selected parent population. Selection on DP is able to leave the noisy area, but then gets stuck in a region with relatively low DP scores.

Ensemble of Local Model Stacks

Thus, instead of using a single local model, DP is calculated using multiple overlapping local models, a *local model stack*. To obtain a *global model stack*, we extend on the ensemble of local models used for PE and PV, and generate an *ensemble of local model stacks*, see Figure 4.10.

For a local model stack with index i, the vector

$$\hat{\mathbf{f}}_i(\mathbf{x}) = \left(\hat{f}_{i,1}(\mathbf{x}), \ldots, \hat{f}_{i,\dim\left(\hat{\mathbf{f}}_i(\mathbf{x})\right)}(\mathbf{x})\right) \tag{4.12}$$

contains the predictions for $\mathbf{x}$ by its $\dim\left(\hat{\mathbf{f}}_i(\mathbf{x})\right) = 10$ models. DP is reformulated as the *minimum* IQR score resulting from evaluating a solution $\mathbf{x}$ on all local model stacks,

$$\mathrm{DP}(\mathbf{x}) = \min_i \ \mathrm{IQR}\left(\hat{\mathbf{f}}_i(\mathbf{x})\right), \tag{4.13}$$

as the models in the originating stack are likely to have the most-accurate prediction for that solution.

In Figure 4.11, the solutions generated in selecting on DP are displayed, as well as the development of the DP scores. In selecting on DP, the search is able to leave the noisy area but, like with PV, gets stuck in a region with relatively low DP scores. Again, this is likely due to self-adaptation in a dynamic scenario that decreases the search horizon to a point where other areas featuring higher DP are obscured.

4.3 · Interestingness

In selecting on learning-based novelty, solutions with maximum modeling error are promoted as predictor of their to-be-induced learning progress after inclusion in model training. Such an estimator of learning progress, however, does not account for detecting and avoiding notoriously hard regions in the search space that are effectively impossible to model given a certain used modeling approach, for instance because of extreme *non-stationary noise* [**Kruisselbrink, 2012**].

From the results obtained by the learning-based novelty expressions, it can be seen that a better estimator of the learning progress is required. On the one hand, these show attraction to regions with noise, on the other hand, the search is seen to stall in seemingly random locations because of a decreasing search horizon combined with ranking solutions based on dynamic values. Instead of directly selecting on the learning-based novelty that the latest state of the surrogate model gave rise to, we can compare the modeling error at different moments in time during the learning process, see Figure 4.12.

In selecting on interestingness, those solutions are picked that improve or are expected to improve the surrogate model's approximation of the to-be-learned mapping most by including them in model training, after sampling them. This is equivalent to the aim of learning-based novelty, but interestingness includes an additional scheme for detecting or estimating whether a certain region can effectively be modeled.

In maximizing learning progress, the goal is to invoke an optimal sampling pattern for building an accurate model of the search space. Interestingness should help skipping unlearnable areas, but *does not* express the chance of finding high-quality solutions somewhere. An area of the quality landscape featuring higher variation in quality than average should in principle remain interesting longer as it requires a more dense sampling for accurate modeling than a more regular landscape. However, this does not mean that this area is a good candidate for exploitation on quality! The area can be rugged but feature far-from-optimal quality values only. Furthermore, that it

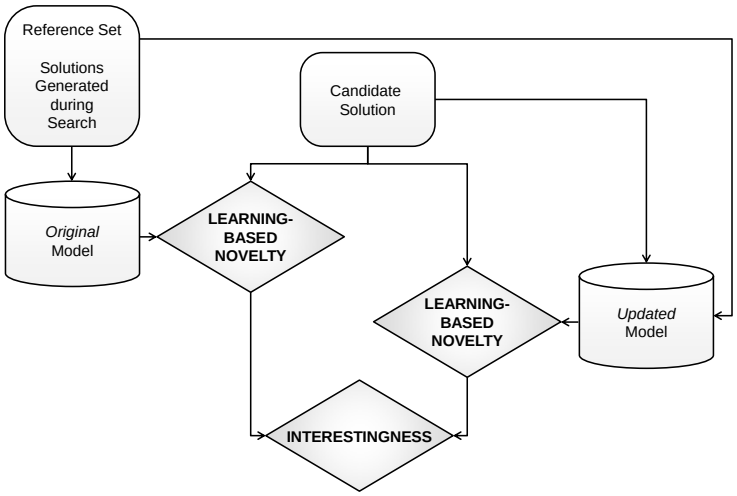


Figure 4.12 Interestingness. Based on the development of the learning-based novelty in the region that a candidate solution lies in, (for instance) determined before and after including it in model training, interestingness expresses the learning progress induced by the candidate solution (or in its region).

remained interesting for a relatively long time could also be due to the fact that it is unlearnable. Our assumption is that exploration based on interestingness assists in efficiently uncovering diverse solutions, as it should move quickly through areas with low information density or with properties making them unlearnable, and thus with high likeliness unexploitable via optimization on quality.

4.3.1 · Notational Scheme

The interestingness measures that are presented in Section 4.3.3 *determine* or *derive* the learning progress that a candidate solution induces after inclusion in model training. For clearly formulating the different interestingness expressions, we present the following general notational scheme:

- t is the current position in time; time is viewed as passing *discretely*, each iteration of the iterative search method defining a single time step; t is initialized at 0;

- $\mathbf{x}_{i,t}$ is a candidate solution from the set X_t of all candidate solutions at time t, with $1 \leq i \leq |X_t|$; notably, X_t contains the solutions that are candidate *at* time t, *not* all previous candidates up to time t;

- $\mathcal{R}(\mathbf{x}_{i,t})$ denotes the *region* in the design space in which $\mathbf{x}_{i,t}$ lies; a definition of what constitutes a region is given in Section 4.3.2;
- $\mathbf{s}_{j,t} = (\mathbf{q}, f(\mathbf{q}))$ is a solution $\mathbf{q}$ sampled at time t, that is, for which quality value $f(\mathbf{q})$ was determined; possibly, multiple solutions are sampled at a single time step t, hence the index j;
- $S_t = \bigcup_j \{\mathbf{s}_{j,t}\}$ is the set of all solutions sampled at time t; not necessarily all candidate solutions at time t are sampled: If the modeling errors are estimated, only a subset of X_t needs to be sampled;
- $\mathcal{M}(T)$ is a surrogate model that approximates the mapping from solution vectors $\mathbf{x}$ to quality values $f(\mathbf{x})$, and that is trained on tuples of solutions with quality values in the supplied training set T;
- $T_t = \bigcup_{k=0}^{t-1} S_k$ is the training data available for $\mathcal{M}$ at time t, containing tuples of the solutions sampled at all earlier time steps with accompanying quality values;
- V_t is the test set for determining the performance of $\mathcal{M}$ at time t; in case of PE, V_t needs to contain tuples of solution vectors with accompanying quality values $(\mathbf{v}, f(\mathbf{v}))$, in case of DP and PV the quality component is not used;
- $\mathrm{ME}(\mathbf{v}, \mathcal{M})$ is the modeling error that $\mathcal{M}$ gives rise to for a solution $(\mathbf{v}, f(\mathbf{v})) \in V_t$ at time t; it is either determined as $\mathrm{PE}(\mathbf{v})$, or approximated as $\mathrm{DP}(\mathbf{v})$ or $\mathrm{PV}(\mathbf{v})$;
- $\mathrm{Err}(\mathcal{R}, \mathcal{M}, V)$ is the *regional error value* for region $\mathcal{R}$ that is derived from the modeling errors $\mathrm{ME}(\mathbf{v}, \mathcal{M})$ for solutions $\mathbf{v} \in V$.

4.3.2 · Regional Error Value

Central to all interestingness measures is the need for determining regional error values from the observed modeling errors. The interestingness measures compare the regional error at two time steps; as such, it needs to be determined which of the tested points at a certain time step belong to a certain region, and multiple modeling errors are to be aggregated into a regional error value.

To come to a definition for the regional error, we start from the idea that the influence of an observed modeling error on neighboring points can be expressed using a Gaussian function. Therefore, instead of exclusively assigning test points $\mathbf{v}$

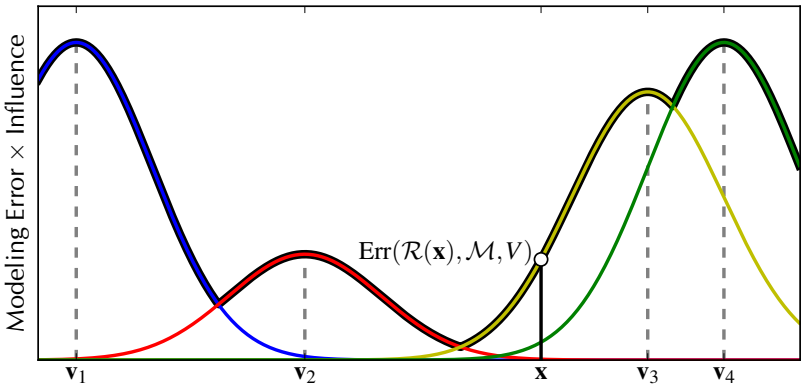


Figure 4.13 Regional Error Value. The regional error $\text{Err}(\mathcal{R}(\mathbf{x}), \mathcal{M}, V)$ in the region $\mathcal{R}(\mathbf{x})$ that is defined by the position of $\mathbf{x}$, is determined by multiplying the modeling errors in the test points $\mathbf{v}$ with an expression of their influence in region $\mathcal{R}(\mathbf{x})$. The influence of a test point in region $\mathcal{R}(\mathbf{x})$ is calculated as $\text{pdf}_{\mathcal{N}(\mathbf{v},\mathbf{I})}(\mathbf{x})$, expressing its influence using a standard normal distribution around the test point $\mathbf{v}$. The maximum of these modeling errors times their influence, at position $\mathbf{x}$, is adopted as the regional error.

to regions, the influence of a test point's modeling error $\text{ME}(\mathbf{v}, \mathcal{M})$ is calculated using a specific Gaussian function, namely, the probability density function of the standard normal distribution $\mathcal{N}(\mathbf{v}, \mathbf{I})$ around $\mathbf{v}$, $\text{pdf}_{\mathcal{N}(\mathbf{v},\mathbf{I})}(\mathbf{x})$. Hence, perhaps a bit contradictory, a region $\mathcal{R}(\mathbf{x})$ is defined merely by the position of $\mathbf{x}$ that indicates the influence of the test points' modeling errors. The regional error value is calculated as the *maximum* of the modeling errors $\text{ME}(\mathbf{v}, \mathcal{M})$ in the points $\mathbf{v}$, from the test set V, multiplied by each point's influence [**Reehuis et al., 2013a**, **Reehuis et al., 2013c**], see Figure 4.13. Thus, we determine the test point $\mathbf{v}$ with the highest resulting modeling error after decreasing it relatively to the distance between $\mathbf{x}$ and $\mathbf{v}$, based on the probability density function of a normal distribution around $\mathbf{v}$,

$$\text{Err}(\mathcal{R}(\mathbf{x}), \mathcal{M}, V) = \max_{\mathbf{v} \in V} \left(\underbrace{\text{ME}(\mathbf{v}, \mathcal{M})}_{\text{modeling error}} \cdot \underbrace{(2\pi)^{-\frac{1}{2}|\mathbf{v}|} \exp\left(-\tfrac{1}{2}(\mathbf{x}-\mathbf{v})'\mathbf{I}(\mathbf{x}-\mathbf{v})\right)}_{\text{influence of } \mathbf{v} \text{ on } \mathbf{x}\text{, calculated as } \text{pdf}_{\mathcal{N}(\mathbf{v},\mathbf{I})}(\mathbf{x})} \right). \quad (4.14)$$

The result is that *all* regions get an error value at each time step for which training data is available, anywhere in the search space, and if no training data is available at time u, the regional error $\text{Err}(\mathcal{R},\ \mathcal{M}(T_u = \emptyset),\ V_u)$ is set to 0.

Most of the interestingness measures in Section 4.3.3 are intended to be calculated by comparing regional error *averages* $\langle \text{Err}(\mathcal{R}, \mathcal{M}, V) \rangle$ over two time *periods* [**Oudeyer et al., 2007**, **Baranes and Oudeyer, 2013**, **Luciw et al., 2011**], instead of regional error

values $\mathrm{Err}(\mathcal{R}, \mathcal{M}, V)$ at two singular time steps. Next, we therefore give an overview of different ways of obtaining two distinct regional error averages.

Singular Errors

Resuming, the first option for obtaining regional errors at time steps u and $u - \tau$ is thus simply adopting the non-averaged, singular error values at those time steps,

$$\langle \mathrm{Err}(\mathcal{R}, \mathcal{M}(T_u), V_u) \rangle \leftarrow \mathrm{Err}(\mathcal{R}, \mathcal{M}(T_u), V_u), \tag{4.15}$$

where the test sets V_u and $V_{u-\tau}$ are defined by the used interestingness measure.

Separated Averages

[**Oudeyer et al., 2007**] propose calculating the mean error over two time periods with window size w that lie τ time steps apart. The mean error over the period ending in time step u and over the period ending in $u - \tau$ is calculated as follows,

$$\langle \mathrm{Err}(\mathcal{R}, \mathcal{M}(T_u), V_u) \rangle \leftarrow \frac{1}{w} \sum_{k=0}^{w-1} \mathrm{Err}(\mathcal{R}, \mathcal{M}(T_{u-k}), V_{u-k}). \tag{4.16}$$

Based on [**Oudeyer et al., 2007**], we use $w = \min\left(15, \left\lfloor \frac{1}{2} \cdot u \right\rfloor\right)$ and $\tau = \min(25, u - w)$; w and τ are increased until its listed value has become possible.

Adjacent Averages

In [**Baranes and Oudeyer, 2013**], a simplification of this scheme is proposed through assigning the $2 \cdot w$ most-recent time steps to the two error averages, taking $w = \left\lfloor \frac{1}{2} \cdot \min(40, u) \right\rfloor$ and $\tau = w$.

Short-term/Long-term Memory

[**Luciw et al., 2011**] use *exponential smoothing* to obtain error averages in the form of two error *memories*, one with a slower forgetting rate ($\mathrm{Mem}_{\mathrm{long}}$) than the other ($\mathrm{Mem}_{\mathrm{short}}$),

$$\begin{aligned} \mathrm{Mem}_{\mathrm{long}}(\mathcal{R}, u) &= (1 - \eta_{\mathrm{long}}) \cdot \mathrm{Mem}_{\mathrm{long}}(\mathcal{R}, u - 1) \;+\; \eta_{\mathrm{long}} \cdot \mathrm{Err}(\mathcal{R}, \mathcal{M}(T_u), V_u), \\ \mathrm{Mem}_{\mathrm{short}}(\mathcal{R}, u) &= (1 - \eta_{\mathrm{short}}) \cdot \mathrm{Mem}_{\mathrm{short}}(\mathcal{R}, u - 1) \;+\; \eta_{\mathrm{short}} \cdot \mathrm{Err}(\mathcal{R}, \mathcal{M}(T_u), V_u), \end{aligned} \tag{4.17}$$

using $\eta_{\text{long}} = 0.1$ for forgetting slowly, and $\eta_{\text{short}} = 0.2$ for forgetting quicker (settings from [**Luciw et al., 2011**]). $\text{Mem}_{\text{long}}(\mathcal{R}, u)$ is used as the regional error average at time $u - \tau$, and $\text{Mem}_{\text{short}}(\mathcal{R}, u)$ is used as the regional error average at time u.

4.3.3 · Interestingness Measures

Before we can describe the interestingness measures, a definition of the *observed learning progress* in a region $\mathcal{R}$ at time u is required. This is calculated as the discrete-time derivative of the regional error $\text{Err}(\mathcal{R}, \mathcal{M}, V)$, taken as the decrease of the error from time $u - \tau$ to time u [**Oudeyer and Kaplan, 2007**],

$$\text{LP}(\mathcal{R}, u) = \text{Err}(\mathcal{R}, \mathcal{M}(T_{u-\tau}), V_{u-\tau}) - \text{Err}(\mathcal{R}, \mathcal{M}(T_u), V_u). \tag{4.18}$$

To determine the performance of surrogate model $\mathcal{M}(T_u)$, at each time step u, the set X_u of candidate solutions *at* time u serves as test set V_u, as these are the solutions for which the modeling error was determined,

$$\underbrace{\text{Err}(\mathcal{R}, \mathcal{M}(T_u), V_u = X_u)}_{\text{model performance in } \mathcal{R} \text{ at time } u}. \tag{4.19}$$

Now, importantly, Equation 4.18 provides the learning progress observed in region $\mathcal{R}$ at time u, depending on the modeling errors *observed* in the candidate solutions at time u. However, in order to determine the learning progress that a solution that is candidate at time u itself *induces*, we have to be *one time step in the future*, at time $u + 1$. The model has to be re-trained, now including the candidate solution, and then tested again in the same region.

Therefore, to determine the learning progress that a solution that is candidate at time t will have given rise to at time $t + 1$, the following steps are to be performed:

1. At time $t + 1$, in the region that candidate solution $\mathbf{x}_{i,t}$ defines, determine the previous regional error based on the modeling errors that the model at time t (taking $\tau = 1$) showed for the set X_t of candidate solutions at time t,

$$\text{Err}\left(\mathcal{R}(\mathbf{x}_{i,t}), \mathcal{M}(T_t), V_t = X_t\right); \tag{4.20}$$

2. Re-train the model on the union of the training set used at time t and the candidate solution $\mathbf{x}_{i,t}$, which needs to have been sampled, and again determine the error in region $\mathcal{R}(\mathbf{x}_{i,t})$, now defined as

$$\text{Err}(\mathcal{R}(\mathbf{x}_{i,t}), \mathcal{M}(T_t \cup \{(\mathbf{x}_{i,t}, f(\mathbf{x}_{i,t}))\}), V_{t+1} = X_{t+1}), \tag{4.21}$$

thus, the regional error at time $t+1$ is based on the modeling errors observed in the new set of candidate solutions X_{t+1};

3. By subtracting the regional error at time t and $t+1$ according to Equation 4.18, we obtain the learning progress that $\mathbf{x}_{i,t}$ has induced.

For sake of completeness, it is pointed out that in case of using PE as modeling error within the regional error calculation, all candidate solutions are evaluated on quality. This is needed for determining their PE and later on for re-training the model. In estimating the modeling error using DP or PV, only the selected promising candidate solutions are sampled, needed for re-training the model.

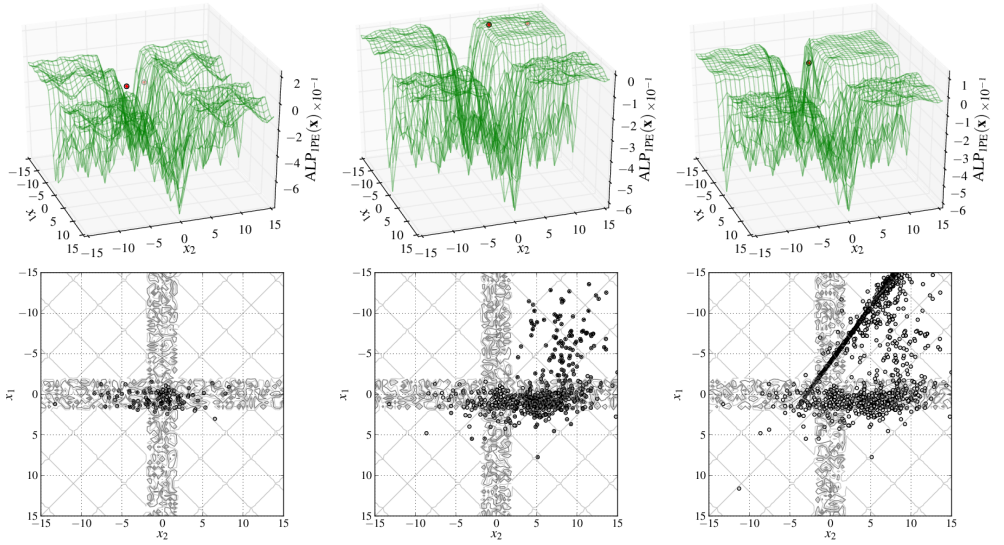
4.3.3.1 · Actual Learning Progress

Schmidhuber [**Schmidhuber, 1991**, **Oudeyer and Kaplan, 2007**, **Oudeyer et al., 2007**] follows the steps laid out above, but at time t already re-trains the model, for *each* candidate solution $\mathbf{x}_{i,t}$, thus generating $|X_t|$ temporary[3] models to be able to express each candidate's interestingness. The requirement of being one time step in the future rises from the fact that a new set of candidate solutions X_{t+1} is to be used in evaluating the updated model. This is, however, circumvented by evaluating the updated model on the *same* test set again, namely the candidate solutions at time t. We term this measure *actual learning progress* (ALP),

$$\mathrm{ALP}(\mathbf{x}_{i,t}) = \underbrace{\mathrm{Err}\left(\mathcal{R}(\mathbf{x}_{i,t}), \mathcal{M}(T_t), V_t = X_t\right)}_{\text{error in region } \mathcal{R}(\mathbf{x}_{i,t}) \text{ before training on } \mathbf{x}_{i,t}} - \underbrace{\mathrm{Err}\left(\mathcal{R}(\mathbf{x}_{i,t}), \mathcal{M}(T_t \cup \{(\mathbf{x}_{i,t}, f(\mathbf{x}_{i,t}))\}), V_{t+1} = X_t\right)}_{\text{error in region } \mathcal{R}(\mathbf{x}_{i,t}) \text{ after training on } \mathbf{x}_{i,t}}. \tag{4.22}$$

In Figure 4.14, the solutions generated in selecting on $\mathrm{ALP}_{1\mathrm{PE}}$ are displayed, as well as the development of the $\mathrm{ALP}_{1\mathrm{PE}}$ scores. 1PE stands for singular regional error values based on PE. Single time steps are used, as ALP compares the error directly before and after training on a candidate, and PE is used because of higher reliability compared to estimating. In selecting on $\mathrm{ALP}_{1\mathrm{PE}}$, the search is able to move through the unlearnable area, and exploration does not get stuck after. The unlearnability of

[3] It could be argued that only $|X_t|-1$ models are temporary, as the model of the best-ranked candidate can be preserved, but this does not hold in case multiple candidates are selected to proceed to the next iteration, requiring training a new model on all of these.



(a) Generation 20 **(b)** Generation 100 **(c)** Generation 495

Figure 4.14 Actual Learning Progress. The bottom-row plots show the solutions generated by a single run on test problem SineUnlearnable, of a (3,6)-CMA-ES selecting on ALP_{1PE}, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored). The top-row plots display the development of the ALP_{1PE} scores, where the red dots represent the lastly-selected parent population. The search is able to traverse the noisy area and keeps exploring after.

the region with extreme noise is cleary reflected in negative ALP_{1PE} scores, while the "sine"-areas get slightly positive scores. ALP is computationally costly, however, because of the need of training a temporary model for each candidate solution.

4.3.3.2 · Previous Learning Progress

Oudeyer et al. propose in [**Oudeyer et al., 2007**, **Oudeyer and Kaplan, 2007**] for each region $\mathcal{R}$ to adopt the learning progress $\text{LP}(\mathcal{R}, \theta_{\mathcal{R}})$ that was realized at the last time step $\theta_{\mathcal{R}}$ at which a solution was generated in that region. For the used regional error value definition, it holds that, at time t, for all regions $\theta_{\mathcal{R}}$ equals t.

The *previous learning progress* (PLP) measure indicates for a region whether it has been possible to better approximate the to-be-learned mapping there at time step t, and should thereby detect regions in which the surrogate model does not improve. For a candidate $\mathbf{x}_{i,t}$, PLP is taken as the learning progress that *was* realized in the region $\mathcal{R}(\mathbf{x}_{i,t})$ that it defines, at time t (candidate $\mathbf{x}_{i,t}$ itself induces learning progress at time

$t + 1$),

$$\mathrm{PLP}(\mathbf{x}_{i,t}) \;=\; \mathrm{LP}(\mathcal{R}(\mathbf{x}_{i,t}), t) \;=$$

$$\underbrace{\mathrm{Err}\left(\mathcal{R}(\mathbf{x}_{i,t}), \mathcal{M}(T_{t-\tau}), V_{t-\tau} = X_{t-\tau}\right)}_{\text{error in region } \mathcal{R}(\mathbf{x}_{i,t}) \text{ at time } \underline{t-\tau}} - \underbrace{\mathrm{Err}\left(\mathcal{R}(\mathbf{x}_{i,t}), \mathcal{M}(T_t), V_t = X_t\right)}_{\text{error in region } \mathcal{R}(\mathbf{x}_{i,t}) \text{ at time } \underline{t}}. \tag{4.23}$$

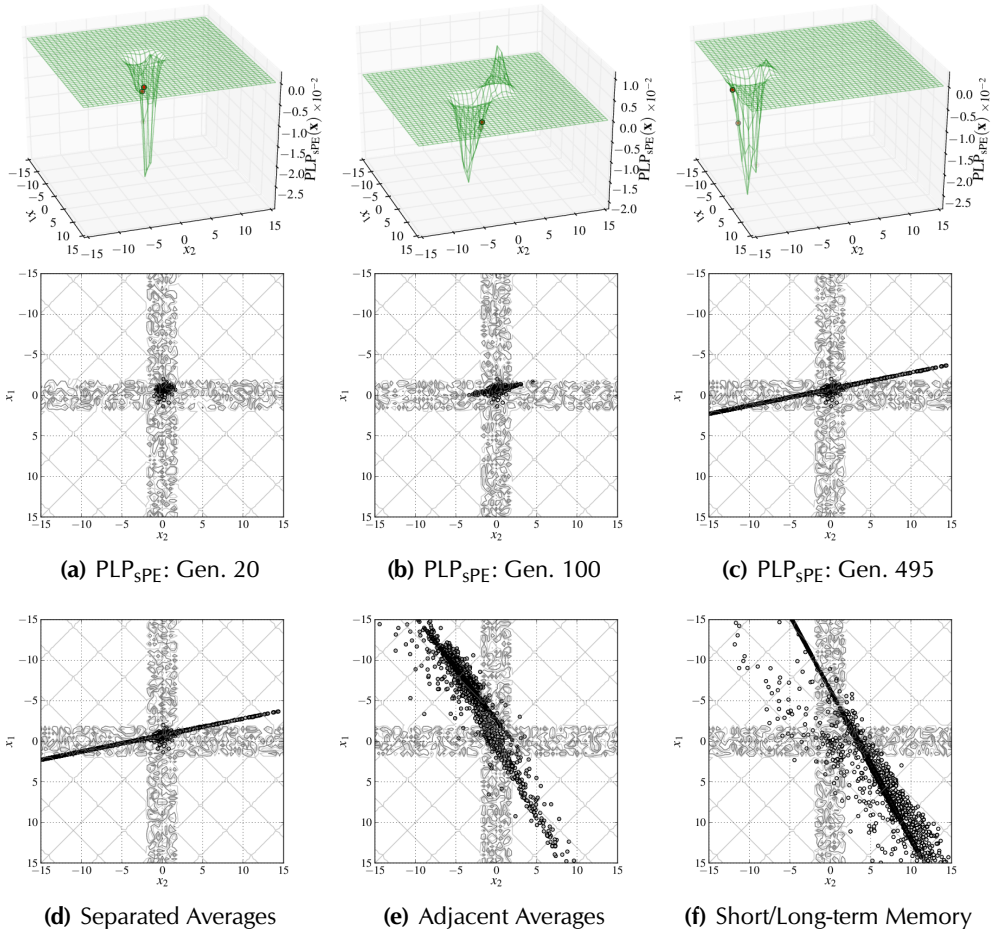
Note that in combination with the used region definition, the discriminative power between candidate solutions $\mathbf{x}_{i,t}$ is limited to the difference in regional error values that arises from the *distance* between the defined regions $\mathcal{R}(\mathbf{x}_{i,t})$.

In Figure 4.15, the solutions generated in selecting on $\mathrm{PLP}_{\mathrm{sPE}}$ are displayed, the development of the $\mathrm{PLP}_{\mathrm{sPE}}$ scores, and a comparison of all solutions generated using the different regional error averaging schemes. sPE stands for separated averages of regional errors, as in [**Oudeyer et al., 2007**], based on PE because of higher reliability compared to estimating the modeling error.

First of all, it can be seen from the plots of the $\mathrm{PLP}_{\mathrm{sPE}}$ scores that *all* candidate solutions get negative values, as it is the selected best values that are displayed and these are all negative. This would make sense as we are in the unlearnable area, however, there *is* an adjacent peak of positive scores. From the plots of all generated solutions for the different averaging schemes, it can be seen that exploration is moving back-and-forth in the same linear direction, seemingly in pursuit of the positive peak.

Remember, PLP expresses the learning progress realized around a candidate solution based on solutions that were generated in that region at earlier time steps. However, in the used regional error calculation, the less-visited a region is, the closer to zero its regional error value will be! Thus, as is moved into a certain area, the regional error values will *increase* at first, leading to *negative* PLP! Only after the regional error values have reached a sufficient level of reliability, based on a sufficient number of test points in the region, PLP values become positive. This is exactly the positive peak that is seen in the wake of the exploration.

PLP requires a different regional error scheme with a high initialization of regional errors of unvisited regions. In [**Oudeyer et al., 2007**], this is taken care of through starting from a *single* region that gets recursively split as exploration continues, based on some splitting condition to be satisfied. They use a threshold of a maximum number



(a) PLP_{sPE}: Gen. 20 **(b)** PLP_{sPE}: Gen. 100 **(c)** PLP_{sPE}: Gen. 495

(d) Separated Averages **(e)** Adjacent Averages **(f)** Short/Long-term Memory

Figure 4.15 Previous Learning Progress. In (a) through (c), the plots show the solutions generated by a single run on test problem SineUnlearnable, of a (3,6)-CMA-ES selecting on PLP_{sPE}, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored); the top-row plots display the development of the PLP_{sPE} scores, where the red dots represent the lastly-selected parent population. In (d) through (f), the solutions generated in selecting on PE-based PLP using different error averaging schemes are displayed. In all tested setups, selecting on PLP moves back-and-forth in a linear direction because of incompatibility with the used regional error calculation regarding unvisited areas.

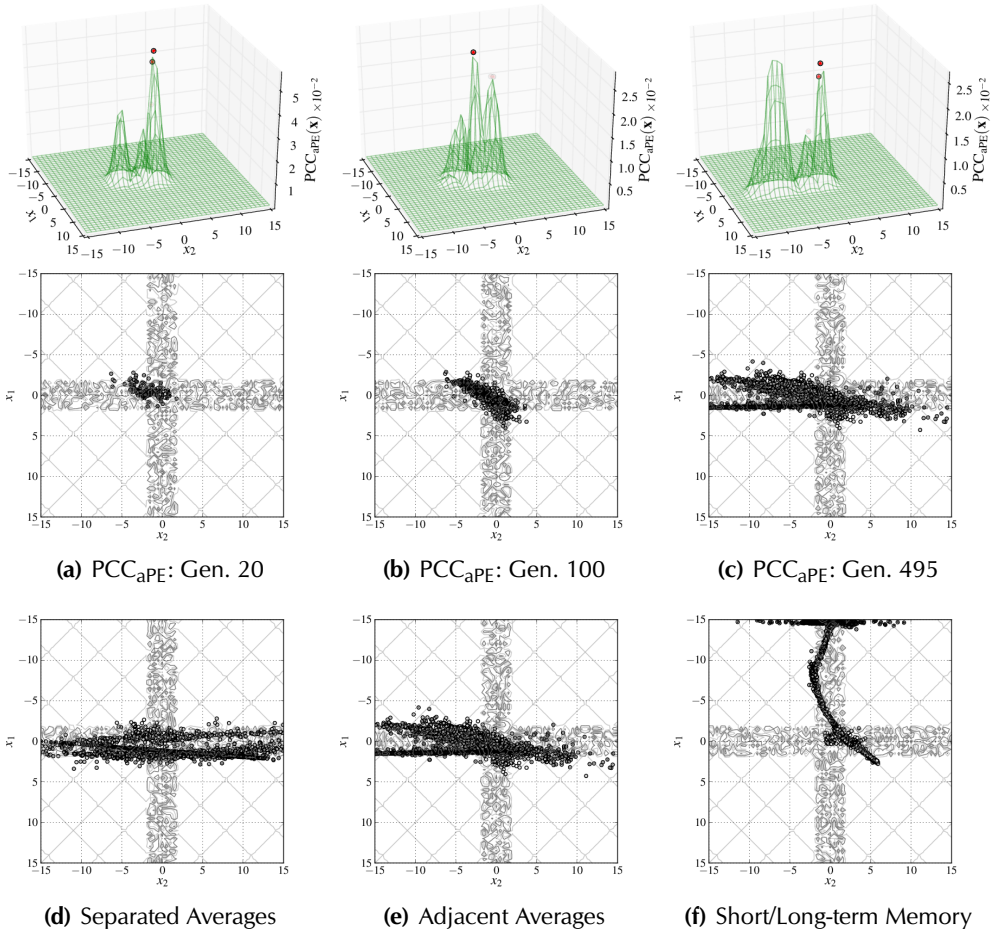
of solutions or test points that may be assigned to a region. When met, the region is split in two based on some clustering technique that minimizes the distance between the solutions in both clusters.

4.3.3.3 · Previous Competence Change

Next to promoting high learning progress, which reflects that the model is capable of improving its mapping in a certain region, [**Baranes and Oudeyer, 2013**] suggest valuing *negative* learning progress, as this could point to deterioration in a region in which improvement was shown earlier. That region should be focused on again for verifying the fact of (not) being able to improve there. *Previous competence change* (PCC) is, like PLP, based on the most-recent learning progress shown in a region, but taking the absolute value,

$$\begin{aligned} \mathrm{PCC}(\mathbf{x}_{i,t}) &= |\mathrm{PLP}(\mathbf{x}_{i,t})| = \\ &|\underbrace{\mathrm{Err}\left(\mathcal{R}(\mathbf{x}_{i,t}), \mathcal{M}(T_{t-\tau}), V_{t-\tau} = X_{t-\tau}\right)}_{\text{error in region } \mathcal{R}(\mathbf{x}_{i,t}) \text{ at time } \underline{t-\tau}} - \\ &\underbrace{\mathrm{Err}\left(\mathcal{R}(\mathbf{x}_{i,t}), \mathcal{M}(T_t), V_t = X_t\right)}_{\text{error in region } \mathcal{R}(\mathbf{x}_{i,t}) \text{ at time } \underline{t}}|. \end{aligned} \tag{4.24}$$

In Figure 4.16, the solutions generated in selecting on $\mathrm{PCC}_{\mathrm{aPE}}$ are displayed, the development of the $\mathrm{PCC}_{\mathrm{aPE}}$ scores, and a comparison of all solutions generated using different regional error averaging schemes. aPE stands for adjacent averages of regional errors, as in [**Baranes and Oudeyer, 2013**], based on PE because of higher reliability compared to estimating the modeling error. Through taking the absolute value, PCC omits some of the problems of PLP with the used regional error scheme. Instead of negative values, little-visited areas get positive values. However, the problem of the trailing positive peak persists, and at times exploration will be drawn back to where it was before. For two of the error averaging schemes, selecting on PCC can be seen mostly returning to previously visited areas, but for short-term/long-term memory more directed behavior is shown.



(a) PCC_{aPE}: Gen. 20 **(b)** PCC_{aPE}: Gen. 100 **(c)** PCC_{aPE}: Gen. 495

(d) Separated Averages **(e)** Adjacent Averages **(f)** Short/Long-term Memory

Figure 4.16 Previous Competence Change. In (a) through (c), the plots show the solutions generated by a single run on test problem SineUnlearnable, of a (3,6)-CMA-ES selecting on PCC_{aPE}, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored); the top-row plots display the development of the PCC_{aPE} scores, where the red dots represent the lastly-selected parent population. In (d) through (f), the solutions generated in selecting on PE-based PCC using different error averaging schemes are displayed. While for separated and adjacent averaging of regional errors, selecting on PCC mostly returns to previously visited areas, more directed exploration can be seen for short-term/long-term memory.

4.3.3.4 · Reducible Error

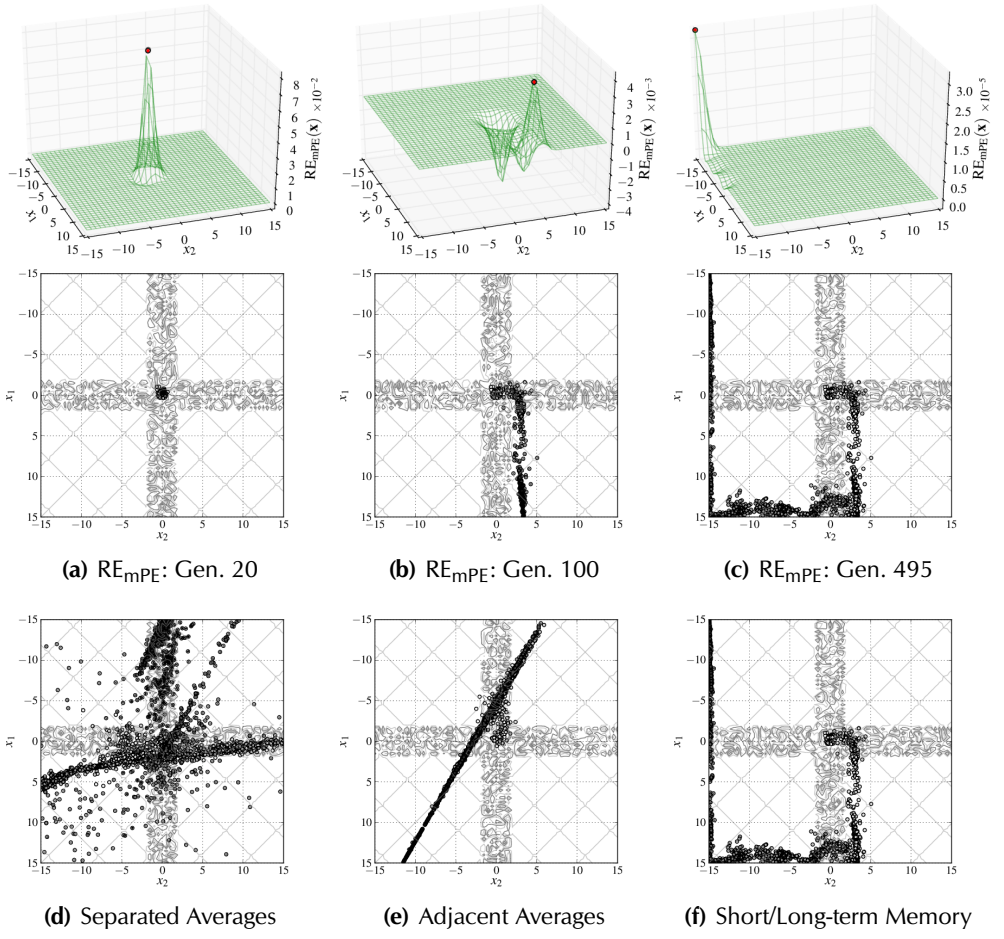
The approach remaining is valuing *only* negative PLP. [**Luciw et al., 2011**] aim for error maximization, like in selecting on learning-based novelty directly, but account for regions that cannot be modeled. They subtract the *irreducible* error, that is, the error in the region at an earlier time step, from the novelty score, resulting in the *reducible error* (RE),

$$\begin{aligned} \mathrm{RE}(\mathbf{x}_{i,t}) \;=\; -\,\mathrm{PLP}(\mathbf{x}_{i,t}) \;=\; & \\ & \underbrace{\mathrm{Err}(\mathcal{R}(\mathbf{x}_{i,t}), \mathcal{M}(T_t), V_t = X_t)}_{\underline{\text{current}}\text{ error in }\mathcal{R}(\mathbf{x}_{i,t}),\text{ at time }\underline{t}} \;- \\ & \underbrace{\mathrm{Err}(\mathcal{R}(\mathbf{x}_{i,t}), \mathcal{M}(T_{t-\tau}), V_{t-\tau} = X_{t-\tau})}_{\underline{\text{irreducible}}\text{ error in }\mathcal{R}(\mathbf{x}_{i,t}),\text{ at time }\underline{t-\tau}}\,. \end{aligned} \tag{4.25}$$

Regions with a high error in the past but a lower error now, *exactly* those that are promoted by PLP, get a *negative* reducible error! These regions have shown learning progress, meaning that the model was improved there *already*, and therefore the model is now expected to be improved more in less-visited regions.

The irreducible error has to be initialized to a low value for new regions that have not been visited yet for the principle to work, not surprisingly, opposite to the requirement for PLP. As stated before, in the used regional error calculation, this is accounted for as regions that have not been frequently visited yet get errors near zero. A region will then feature *high reducible error* upon first visit: High *current* error, and low past, *irreducible* error. In *unlearnable* regions, the current error will remain high, making that the past error becomes high as well after repeated visits, resulting in a *low reducible error*.

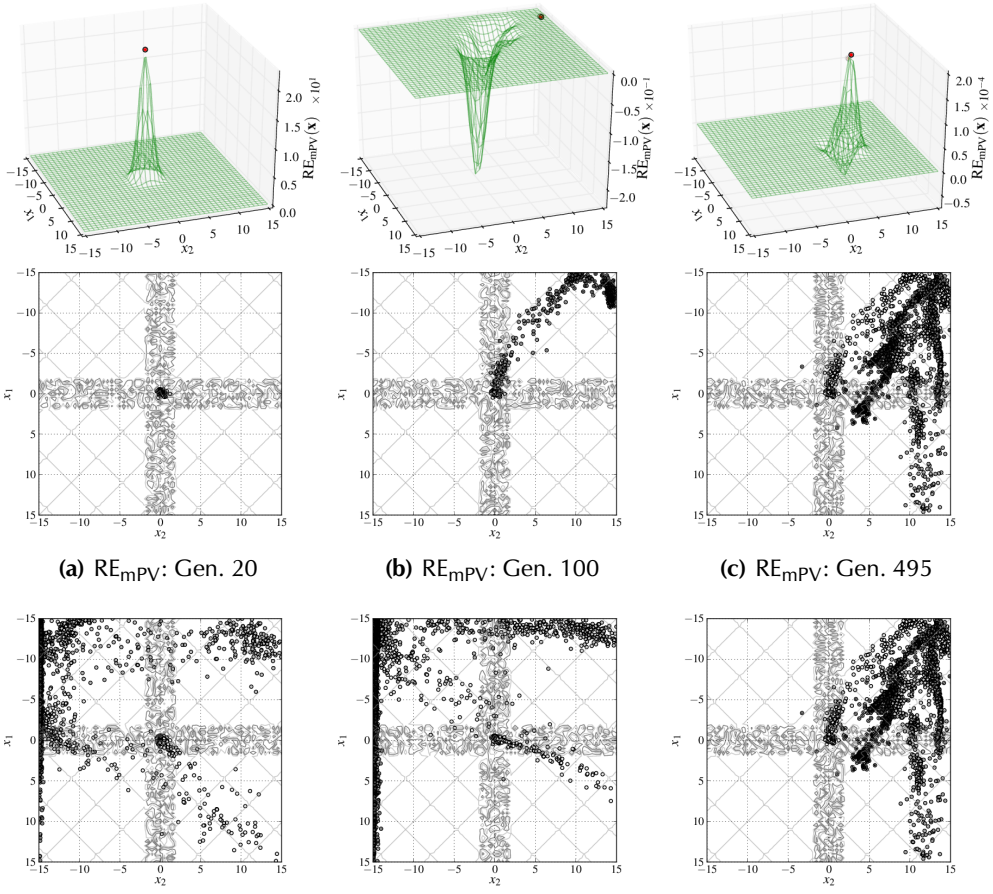
In Figure 4.17, the solutions generated in selecting on $\mathrm{RE}_{\mathrm{mPE}}$ are displayed, the development of the $\mathrm{RE}_{\mathrm{mPE}}$ scores, and a comparison of all solutions generated using different regional error averaging schemes. mPE stands for PE-based long-term/short-term memories of regional errors, as in [**Luciw et al., 2011**]. As aimed for, newly-visited regions and thereby new candidate solutions get high $\mathrm{RE}_{\mathrm{mPE}}$ scores. Furthermore, the trailing peak of high scores reported in PLP and PCC has been inverted to negative. As such, in selecting on $\mathrm{RE}_{\mathrm{mPE}}$, exploration is able to move through the search space in a clearly directed fashion towards regions not visited recently. It should be noted though that this only holds for the memories scheme.



(a) RE_{mPE}: Gen. 20 **(b)** RE_{mPE}: Gen. 100 **(c)** RE_{mPE}: Gen. 495

(d) Separated Averages **(e)** Adjacent Averages **(f)** Short/Long-term Memory

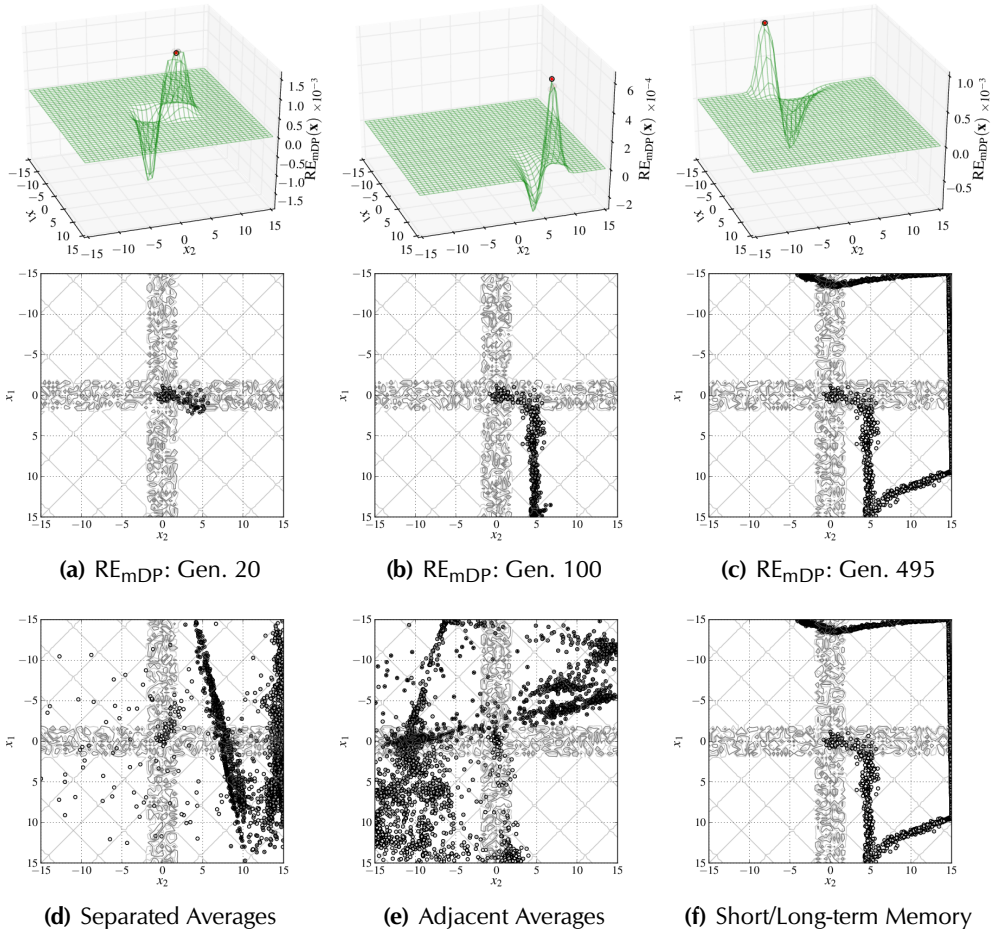
Figure 4.17 Reducible Error — PE. In (a) through (c), the plots show the solutions generated by a single run on test problem SineUnlearnable, of a (3,6)-CMA-ES selecting on RE_{mPE}, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored); the top-row plots display the development of the RE_{mPE} scores, where the red dots represent the lastly-selected parent population. In (d) through (f), the solutions generated in selecting on PE-based RE using different error averaging schemes are displayed. In selecting on RE, exploration is able to move in a clearly directed pattern towards unvisited areas. This is, however, only the case in using short-term/long-term memories for regional error averaging.



(a) RE_{mPV}: Gen. 20 **(b)** RE_{mPV}: Gen. 100 **(c)** RE_{mPV}: Gen. 495

(d) Separated Averages **(e)** Adjacent Averages **(f)** Short/Long-term Memory

Figure 4.18 Reducible Error — PV. In (a) through (c), the plots show the solutions generated by a single run on test problem SineUnlearnable, of a (3,6)-CMA-ES selecting on RE_{mPV}, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored); the top-row plots display the development of the RE_{mPV} scores, where the red dots represent the lastly-selected parent population. In (d) through (f), the solutions generated in selecting on PV-based RE using different error averaging schemes are displayed. In selecting on PV-based RE, exploration is not able to match the performance of PE-based RE.



(a) RE_{mDP}: Gen. 20 **(b)** RE_{mDP}: Gen. 100 **(c)** RE_{mDP}: Gen. 495

(d) Separated Averages **(e)** Adjacent Averages **(f)** Short/Long-term Memory

Figure 4.19 Reducible Error — DP. In (a) through (c), the plots show the solutions generated by a single run on test problem SineUnlearnable, of a (3,6)-CMA-ES selecting on RE_{mDP}, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored); the top-row plots display the development of the RE_{mDP} scores, where the red dots represent the lastly-selected parent population. In (d) through (f), the solutions generated in selecting on DP-based RE using different error averaging schemes are displayed. While using only half of the quality evaluations for model training compared to PE-based RE, DP-based RE is able to deliver the same performance, for error averaging based on short-term/long-term memory.

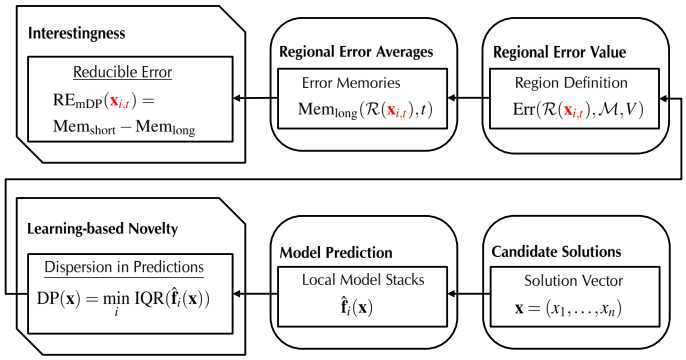


Figure 4.20 RE–mDP Summarized. Starting from the *learning-based novelty* of solutions at their time of candidacy, these DP scores are aggregated (*regional error value*) depending on the proximity to the point $\mathbf{x}_{i,t}$ for which *interestingness* is to be evaluated. Averaged over time (*regional error averages*), a high current error (*short-term memory*) points to possible model improvement and thus at possibly realizing *learning progress*. Unlearnable regions are accounted for through subtracting the *irreducible error*, in the form of a slower-dissipating memory of regional error values (*long-term memory*), resulting in the *reducible error*.

Motivated by the possibility of matching this performance at lower cost of quality function evaluations, RE is tested with estimating the modeling errors via PV and DP. These results are displayed in Figure 4.18 and 4.19. In selecting on PV-based RE, exploration is not able to induce a similarly-clear pattern of directed behavior, even in using the memories scheme. As analyzed before, PV-based ranking is hindered through the model-wide stationarity, which makes the ranking purely density-based. Moreover, it is likely further hindered by the non-stationarity between different models in the used horizontal ensemble.

DP-based RE, conversely, *is* able to reproduce the PE-based behavior for the memories scheme, with no concessions to performance. Importantly, calculating DP is free of quality evaluations, but moreover, in DP, only half of the solutions are used for model training compared to in PE. Only the selected 3 parent individuals are evaluated and included in model training versus all 6 offspring. This makes $\mathrm{RE}_{\mathrm{mDP}}$ the most-promising learning-based method, which is summarized in Figure 4.20.

4.4 · Summary

In this chapter, in search for exploration criteria for driving design-space exploration, is turned to work describing the use of *novelty* and *interestingness* measures for exploration in *active learning*, in developmental robotics. At the heart of these measures

Table 4.1 Taxonomy of Exploration Measures. Overview of the exploration measures for adaptive search that are described in this chapter. All exploration measures are to be maximized, i.e., a solution with the highest exploration-measure outcome gets the best ranking.

Distance-based Novelty			Distance to reference solutions in an archive, with respect to distance measure d.	Section 4.2.1
Sparseness	$\mathrm{Sp}(\mathbf{x})$	$= \frac{1}{k}\sum_{i=1}^{k} \mathrm{d}(\mathbf{x}, \mathbf{q}_i)$	Average distance to the k closest solutions $\mathbf{q}_i$ from archive S and the current population; Sp has a lot of parameters that need to be set.	Section 4.2.1.1
Uniqueness	$\mathrm{Un}(\mathbf{x})$	$= \min_{(\mathbf{q}, f(\mathbf{q}))\in S} \mathrm{d}(\mathbf{x}, \mathbf{q})$	Minimum distance to a solution $\mathbf{q}$ from archive S; simpler scheme, parameter-free.	Section 4.2.1.2
Learning-based Novelty	$\mathrm{ME}(\mathbf{v}, \mathcal{M}(T))$		Error that model $\mathcal{M}$, trained on set of solutions T, makes in its prediction for solution $\mathbf{v}$; estimator of the learning progress in re-training on $\mathbf{v}$.	Section 4.2.2
Prediction Error	$\mathrm{PE}(\mathbf{x})$	$= \left\lvert \hat{f}(\mathbf{x}) - f(\mathbf{x}) \right\rvert$	The actual ME, i.e., difference between predicted and actual quality value; requires evaluating quality.	Section 4.2.2.1
Predictive Variance	$\mathrm{PV}(\mathbf{x})$	$= s^2(\mathbf{x})$	Approximation of ME as available in Kriging; however, PV is (fully) distance-based.	Section 4.2.2.2
Dispersion in Predictions	$\mathrm{DP}(\mathbf{x})$	$= \mathrm{IQR}\left(\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_{\dim(\hat{\mathbf{f}}(\mathbf{x}))}(\mathbf{x})\right)$	Approximation of ME by comparing predictions of multiple models; evaluation-free replacement for PE.	Section 4.2.2.3
Interestingness	$\mathrm{LP}(\mathcal{R}, u)$ $\mathrm{Err}(\mathcal{R}, \mathcal{M}, V)$	$= \mathrm{Err}(\mathcal{R}, \mathcal{M}(T_{u-\tau}), V_{u-\tau}) - \mathrm{Err}(\mathcal{R}, \mathcal{M}(T_u), V_u)$, $= \max_{\mathbf{v}\in V} (\mathrm{ME}(\mathbf{v}, \mathcal{M}) \times \text{influence of } \mathbf{v} \text{ in } \mathcal{R})$	Better estimator of learning progress, by determining the development of the modeling errors in region $\mathcal{R}$; regional error Err is obtained from the observed MEs.	Section 4.3
Actual Learning Progress	$\mathrm{ALP}(\mathbf{x}_{i,t})$	$= \mathrm{Err}(\mathcal{R}(\mathbf{x}_{i,t}), \mathcal{M}(T_t), V_t) - \mathrm{Err}(\mathcal{R}(\mathbf{x}_{i,t}), \mathcal{M}(T_t \cup \{(\mathbf{x}_{i,t}, f(\mathbf{x}_{i,t}))\}), V_t)$	Decrease of Err in $\mathcal{R}(\mathbf{x}_{i,t})$ after training on $\mathbf{x}_{i,t}$, $\mathcal{R}(\mathbf{x}_{i,t})$ is region around $\mathbf{x}_{i,t}$; computat. intensive.	Section 4.3.3.1
Previous Learning Progress	$\mathrm{PLP}(\mathbf{x}_{i,t})$	$= \mathrm{LP}(\mathcal{R}(\mathbf{x}_{i,t}), t)$	LP in $\mathcal{R}(\mathbf{x}_{i,t})$ *prior* to (possible) training on $\mathbf{x}_{i,t}$; requires different Err calculation, with high init.	Section 4.3.3.2
Previous Competence Change	$\mathrm{PCC}(\mathbf{x}_{i,t})$	$= \lvert\mathrm{PLP}(\mathbf{x}_{i,t})\rvert$	PLP and RE combined, less sensitive to Err calc.	Section 4.3.3.3
Reducible Error	$\mathrm{RE}(\mathbf{x}_{i,t})$	$= -\mathrm{PLP}(\mathbf{x}_{i,t}) = -\mathrm{LP}(\mathcal{R}(\mathbf{x}_{i,t}), t)$	Subtract past Err at $t-\tau$ from current Err at t, which omits regions where Err stays high; needs Err calc. initialization at 0; best performing measure.	Section 4.3.3.4

is a *reference set* of solutions that is updated during the search, making it possible to induce a sequential pattern in which exploration is first drawn to certain areas and then repelled again. *Distance-based* novelty compares candidate solutions to the solutions contained in the reference set using a *domain-specific distance measure*, whereas *learning-based* novelty expresses the error that a model generated based on the reference set makes in its approximation for a candidate solution.

The goal of the exploration is finding a *diverse* set of solutions with respect to the same domain-specific distance measure that is used in distance-based novelty. As such, the pattern of exploration induced by selecting on distance-based novelty will automatically constitute a diverse set. The optimal set of solutions for generating an accurate model of the search space is likely to, implicitly, maximize diversity as well. Learning-based novelty as direct predictor of a model's gain of represented knowledge, however, is not sufficient. For inducing a pattern of exploration that maximizes this *learning progress*, regions in which the model does not improve are to be detected to prevent exploration of getting stuck in these areas. This is accounted for in interestingness, which derives from the development of learning-based novelty in a region over time.

This chapter identified two promising exploration criteria, out of nine examined, see Table 4.1, to be integrated into quality-based optimization. *Uniqueness*, a distance-based novelty expression, preserves the explorative behavior of the original *sparseness* expression but with less parameters involved, and leads to quick, oscillating behavior that generates a diverse set of solutions. Interestingness expression *reducible error*, based on short-term/long-term memories of *dispersion-in-predictions* estimations of learning-based novelty, matches the performance of its counterpart based on *prediction-error* learning-based novelty. It leads to directed behavior that generates a clear path through the search space, while managing to do so at half of the cost of the quality function evaluations used for model training compared to the prediction-error-based variant.

5

Integration of Exploration Criteria in the Search

In the previous chapter, criteria for inducing exploration in a search space were examined. In that first step, the resulting sampling behavior of selecting on such exploration criteria was studied, without including quality as primary objective. This chapter aims to conciliate quality, i.e., the exploitation criterion, and the exploration criterion in the search, minimizing conflicts between them and maximizing their combined performance, evaluated as the ability to find a varied set of high-performing solutions.

Basically, this means composing an appropriate *selection scheme* that takes both criteria into account. The exploration criterion has to be enabled to efficiently direct exploitation to new areas. The exploration criteria that are considered are dynamic, depending on the solutions encountered earlier during the search. This makes the search a sequential endeavor that requires, implicitly or explicitly, switching between phases of exploration and exploitation. While combinations of exploration values can be used (see, e.g., [**Hester and Stone, 2012**]), we look for a scheme combining a single exploration criterion and quality.

Niching

Niching methods (see, e.g., [**Shir et al., 2010**]) are selection schemes that do not apply an exploration criterion, in our sense of the term, but that are aimed at actively promoting diversity in their result set. A niching approach runs on top of a search algorithm and makes it target different optima in parallel, performing parallel optimizations in different *niches* in the search space. Niching is able to do so through assuming that

a certain *minimum distance* exists between the different optima. This *niching radius* plays a central role, a proper estimate of it is required for the process to work.

In [**Shir et al., 2010**], advanced niching approaches are described, intended specifically to run on top of the CMA-ES (see Chapter 2) and divided in *fixed-radius* and *adaptive-radius* schemes. Importantly, all these schemes are based on distance between solutions in the *decision space*, that is, based on comparing solution vectors directly. This allows to couple the development of the niching radius to the development of the search distribution in the CMA-ES, leading to adaptive-radius schemes, and to provide a general rule for initializing a fixed radius, but these options are not available in using an arbitrary distance measure.

As described in Chapter 3, a *domain-specific distance measure* will be used for valuing the diversity in found solutions, for evaluating the performance of a search method. If we want to use niching to directly promote diversity with respect to an arbitrary domain-specific distance measure, we are limited to fixed-radius niching. A fixed minimum distance has to be provided, quantified in domain-specific distance, that will be kept between different parallel optimizations. However, if sensibly can be assumed that distance in the decision space reflects domain-specific distance, the advanced niching approaches can be used *as is* to *implicitly* improve diversity with respect to the domain-specific distance measure.

Nevertheless, another aspect that further limits the applicability of niching is the need to start the search from a single fixed solution, further elaborated on in the following section. Typically, in using niching, the search is initialized widely in the search space, increasing the chance of identifying the requested number of niches from the start on. If less than the requested number of niches are available at any point during the optimization, niching explores for additional niches by *reinitializing* solutions, i.e., to solution vectors with random values. This exploratory behavior can be strengthened by using the dynamic exploration criteria studied in Chapter 4 to explore for new niches. In such a hybrid scheme, however, the added value of the parallel optimizations is doubtful, as is relied on sequential exploration to be able to start these optimizations. In this chapter we will therefore address inherently sequential schemes only for integrating the exploration criteria.

5.1 · Integration Schemes for Scenarios with Infeasibility

The aim of the search is to find a diverse set of high-quality solutions, but with a specific application in mind, in the domain of airfoil optimization, see Chapter 6. That application poses difficulties to reach this goal using multiple quality-based optimization runs (i.e., the default approach), as the search needs to be started from an initial solution that is known to be *feasible*. Feasibility of a solution is defined here as not violating structural constraints to such an extent that the solution's quality cannot be approximated through *aerodynamic simulation*. The borders of feasible space are not known in advance, therefore, only a small search distribution around the initial solution is used at the start of the search. This makes an adaptive search method optimizing on quality more likely to converge to suboptimal local optima, close to the initial solution, and thereby different instances of the adaptive search more likely to produce similar results.

Test Setup

We capture some of these properties, namely, infeasible areas and a large amount of local optima, in the to-be-minimized composite test function *AckleyInfeasible*, see Figure 5.1, of which a two-dimensional instance is used with search domain $[-15,15]^n \subseteq \mathbb{R}^n$, $n = 2$,

$$f_{\text{AckleyInfeasible}}(\mathbf{x}) = f_{\text{Ackley}}(\mathbf{x}) + \begin{cases} c_1 \cdot f_{\text{Gaussians}}(\mathbf{x}) & \text{if } f_{\text{Gaussians}}(\mathbf{x}) > 0.25 \\ 0 & \text{otherwise} \end{cases} . \quad (5.1)$$

AckleyInfeasible is a combination of the *Ackley* function [**Ackley, 1987**, **Bäck, 1996**]

$$f_{\text{Ackley}}(\mathbf{x}) = -c_1 \cdot \exp\left(-c_2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(c_3 \cdot x_i)\right) + c_1 + e \quad (5.2)$$

that with $c_1 = 20$, $c_2 = 0.2$, $c_3 = 2\pi$ has a surface covered with local minima in a decreasing trend towards the single global optimum of 0 at $(0,0)$, and the *Gaussians* function [**Graening et al., 2010**, **Reehuis et al., 2011**]

$$f_{\text{Gaussians}}(\mathbf{x}) = \max_{j\in\{1,\ldots,20\}} \left(\exp\left(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right)\right) \quad (5.3)$$

that features 20 Gaussian kernels with centers $\boldsymbol{\mu}_j$ and sharing covariance matrix $\boldsymbol{\Sigma} = 1.6^2 \cdot \mathbf{I}_n$ (i.e., variance times the n-by-n identity matrix). The Gaussian kernels are used to introduce infeasible areas in the search space: We consider a solution to be infeasible

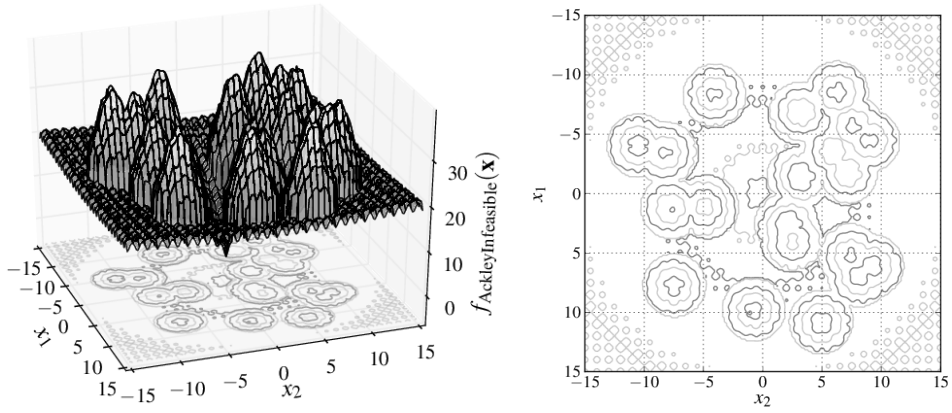


Figure 5.1 Test Function *AckleyInfeasible*. Composite test function based on the Ackley function, including Gaussian kernels for introducing infeasible areas that are modeled as regions with quality values greater than 22. The global minimum is located at (0,0), largely obscured by surrounding infeasible areas.

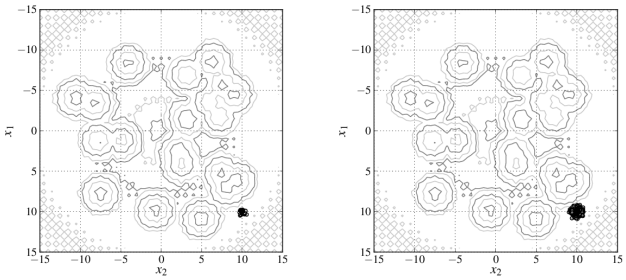
when it has a quality value higher than 22. The kernels have decreasing quality values from the center to the edge, allowing quality-based search to move back to space that is considered feasible. The positions of the kernel centers are chosen randomly in the subdomain $[-10, 10]^n$, but are the same for all function calls.

The search is always initialized in point $(10, 10)$, using a $(6, 12)$-CMA-ES (see Chapter 2) as underlying search method with comma-selection and populations of 6 parents and 12 offspring[1], and an initial stepsize σ_{init} of 0.3. For constraint handling, we add a variable penalty to the criterion that is selected on, i.e., a solution's quality value and/or exploration value, that increases linearly with the extent of violation of the lower boundary of -15 or the upper boundary of 15 per dimension.

Not all integration schemes that are described are considered for analysis of their induced sampling dynamics, only those that we deem promising are tested. Based on the outcome of visual analysis, the performance of certain schemes will be further analyzed using the quality/diversity-Pareto-front interpretation presented in Section 3.2.1, allowing for statistical comparison of attained hypervolume scores. As domain-specific distance measure for test problem AckleyInfeasible, the Euclidean distance between solution vectors is used.

Selected schemes are tested using the two promising exploration criteria found in Chapter 4. These are the distance-based novelty measure *uniqueness* (Un) and

[1] Larger population sizes are used than of the (3,6)-CMA-ES prescribed in Chapter 2 for two-dimensional search spaces, as this is required for the *Pareto-selection* scheme, see Section 5.2.



(a) CMA-ES **(b)** IPOP-CMA-ES

Figure 5.2 Exploration Scheme Required. The solutions generated by a single run on test problem AckleyInfeasible, of a (6,12)-CMA-ES and a (6,12)-IPOP-CMA-ES selecting on quality, are displayed (recently-generated solutions are dark-colored, early-generated solutions are bright-colored). The CMA-ES is run for 495 generations (6000 quality evaluations), and the IPOP-CMA-ES for 105 generations (23k quality evaluations, restarting and doubling the population sizes 5 times). For test function AckleyInfeasible, in combination with an initial stepsize of 0.3, it is clearly visible that an exploration scheme is required for reaching the global minimum at (0,0).

interestingness measure *reducible error* (RE_{mDP}). Interestingness measure RE_{mDP} is based on memories of *dispersion-in-prediction* learning-based novelty scores. For Un and RE_{mDP}, the settings listed in Chapter 4 are used. When an integration scheme requires deviating from these settings, this is described in the section defining it. All methods and techniques used have been implemented using the Shark Machine Learning Library v2.3.43[2].

To illustrate the need for an exploration scheme, we run the CMA-ES and the *IPOP-CMA-ES* [**Auger and Hansen, 2005**] on AckleyInfeasible, see Figure 5.2, using the settings described above. IPOP is a restart scheme that reinitializes with increased population sizes upon meeting an convergence criterion: We use doubling of the population sizes and convergence determined as a stepsize smaller than $0.5 \times \sigma_{\text{init}}$. Both approaches show optimization to (repeatedly, in case of the IPOP-CMA-ES) converge to a local minimum close to the initial point $(10, 10)$.

The remainder of this chapter is organized as follows: In Section 5.2, integration schemes that select on quality and exploration criterion *in parallel* are described. In Section 5.3, a different approach is used: Quality optimization and exploration are *separated* in distinct phases. Section 5.4 provides a summary of the chapter.

[2] http://shark-project.sourceforge.net

5.2 · Multiobjective Selection

A straightforward way of integrating an exploration criterion into quality-based search is taking both criteria into account in parallel. One has the choice between blending or *aggregating* both values into a single fitness value [**Cuccu and Gomez, 2011**, **Cuccu et al., 2011a**] and using a *Pareto-based* multiobjective approach that determines a ranking based on *non-dominance* with respect to both criteria [**Mouret, 2011**, **Graening et al., 2010**]. While being relatively easy to implement, the downside of these schemes is that both objectives can counteract each other.

As described in Chapter 4, our view on exploitation is that of greedy, local search. In exploiting the quality function, the search is therefore best guided by the quality function alone. In zooming in on a local optimum, the search densely samples an increasingly-small area of the search space. In doing so it is obviously hindered by a dynamic exploration criterion that continuously tries to move the search to less explored areas. Conversely, in exploration, the search is hindered by the quality objective in escaping from sufficiently-exploited optima, as their high quality works as an attractor. Nevertheless, as multiobjective schemes are often used, we discuss them here.

5.2.1 · Weighted Aggregation

Aggregating a quality value $f(\mathbf{x})$ and exploration value $\mathrm{EV}(\mathbf{x})$ into a single fitness value is done based on a weighting parameter $0 \leq w \leq 1$ [**Cuccu and Gomez, 2011**, **Cuccu et al., 2011a**],

$$(w-1)\cdot f(\mathbf{x}) + w \cdot \mathrm{EV}(\mathbf{x}). \tag{5.4}$$

In order to make w of influence and mix both criteria meaningfully, both measures have to be normalized to values between 0 and 1 and transformed to be minimized [**Cuccu and Gomez, 2011**]. This can be done by determining the best and worst value (v_{best} and v_{worst}) occurring per criterion in the candidate solutions at each selection round (i.e., the offspring at each generation), and normalizing a value v as follows,

$$v_{\text{norm}} = \frac{v - v_{\text{best}}}{v_{\text{worst}} - v_{\text{best}}}. \tag{5.5}$$

In addition to the hindering effect that both objectives have on each other, choosing an appropriate w-value is no easy task as the optimal setting will change during the search, depending on whether the method is zooming in on an optimum or should

be taken away from it after sufficient exploitation. Thus, ideally, the w-value is set dynamically, updated during the search based on detection of the end of either the exploitation or exploration phase. For instance, if the improvement of the average or best quality occurring in the population falls under a certain magnitude, w can be turned up to escape the current optimum, whereas w can be turned down if the best quality in the population shows improvement over a certain time window, in order to focus the search better [**Cuccu and Gomez, 2011**]. However, exactly *how* to adjust the w-value each time the search should switch between exploitation and exploration remains to be specified.

Relation to Surrogate Utility Functions

Certain utility functions used in surrogate-assisted optimization are actually implementations of weighted aggregation, but they combine *approximated* quality values $\hat{f}(\mathbf{x})$ and associated *variances* $s^2(\mathbf{x})$ instead of actual quality values and general exploration measures; examples are the *lower confidence bound* (LB) and *expected improvement* (ExI) functions, see Section 2.3.1. The central assumption in using these utility functions is that the actual quality value $f(\mathbf{x})$ is a realization of a random variable, for instance, normally-distributed, with mean $\hat{f}(\mathbf{x})$ and variance $s^2(\mathbf{x})$,

$$f(\mathbf{x}) \sim \mathcal{N}\left(\hat{f}(\mathbf{x}), s^2(\mathbf{x})\right). \tag{5.6}$$

An exploration measure is only sensibly applied in a surrogate utility function if it can be used to express the variance of a quality prediction, like in Equation 5.6. We are thus restricted to the model-derived exploration measures, i.e., learning-based novelty and interestingness, see Chapter 4. *Predictive variance* (PV) is by definition suitable, as it is taken equal to the variance $s^2(\mathbf{x})$ that is readily available in certain surrogate modeling approaches, see Section 4.2.2.2. It is commonly used in ExI (e.g., [**Jones et al., 1998, Emmerich et al., 2006**]) and LB (e.g., [**Emmerich et al., 2006, Lu et al., 2013**]). *Prediction error* (PE) requires knowledge of the actual quality value so it does not make sense to apply it in ranking solutions based on predicted quality values. *Dispersion in predictions* (DP) can be used to derive the variance,

$$s^2(\mathbf{x}) \approx \left(\frac{0.68}{0.5} \cdot \mathrm{DP}(\mathbf{x})\right)^2 \tag{5.7}$$

assuming the predictions $\hat{\mathbf{f}}(\mathbf{x})$, see Section 4.2.2.3, to be distributed according to

$$\mathcal{N}\left(\text{median}\left(\hat{\mathbf{f}}(\mathbf{x})\right), s^2(\mathbf{x})\right). \tag{5.8}$$

DP is taken equal to the interquartile range between the predictions $\hat{\mathbf{f}}(\mathbf{x})$, and thus, as it spans two quartiles, represents 50% of the data in the distribution, whereas 68% of the data in the distribution lie within one standard deviation $s(\mathbf{x})$ of the mean according to the 68–95–99.7 rule [**Tanton, 2005**]. The mean is here median $\left(\hat{\mathbf{f}}(\mathbf{x})\right)$.

Interestingness expressed as reducible error is a decreased variant of the current error, a solution's learning-based novelty value, and is obtained by subtracting the *irreducible error* that was determined in the region in which the solution lies, see Section 4.3.3.4. The interestingness measure RE_{mDP} can be used as estimate of the variance in place of DP,

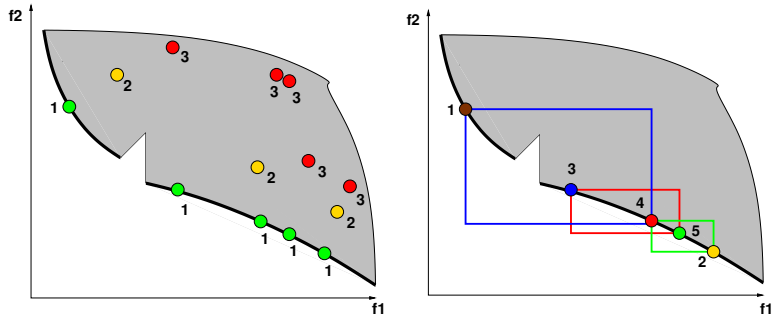
$$s^2(\mathbf{x}) \approx \left(\frac{0.68}{0.5} \cdot \text{RE}_{\text{mDP}}(\mathbf{x})\right)^2. \tag{5.9}$$

RE_{mDP} decreases DP values and thereby the derived variance quantities for those solutions in areas that show little learning progress, leading to worse rankings for these solutions based on the utility functions.

5.2.2 · Pareto-based Selection

In Pareto-based selection, candidate solutions are ranked on both their quality value $f(\mathbf{x})$ and exploration value $\text{EV}(\mathbf{x})$, by iteratively determining *fronts* of *non-dominated* solutions and providing a secondary comparison criterion for solutions on the same front. Restricting the definition to two objectives, a candidate solution $\mathbf{x}$ is non-dominated if there is no other solution $\mathbf{x}'$ in the set of candidate solutions that is better than $\mathbf{x}$ with respect to both objectives simultaneously, or better with respect to one objective and at least equal with respect to the other objective.

Non-dominated sorting partitions the set of candidates into fronts based on domination, where the first front contains all solutions that are non-dominated by all other candidate solutions. The second front is obtained by excluding the solutions from the first front from the set, and then determining the non-dominated set under these conditions, and so forth. As non-dominated sorting results in solutions with the same rank, see Figure 5.3, a second criterion that discriminates between solutions with equal rank based on their placement on the mutual front is required. We use the *crowding-distance* criterion [**Deb et al., 2002**], aimed at evenly distributing points over the front.

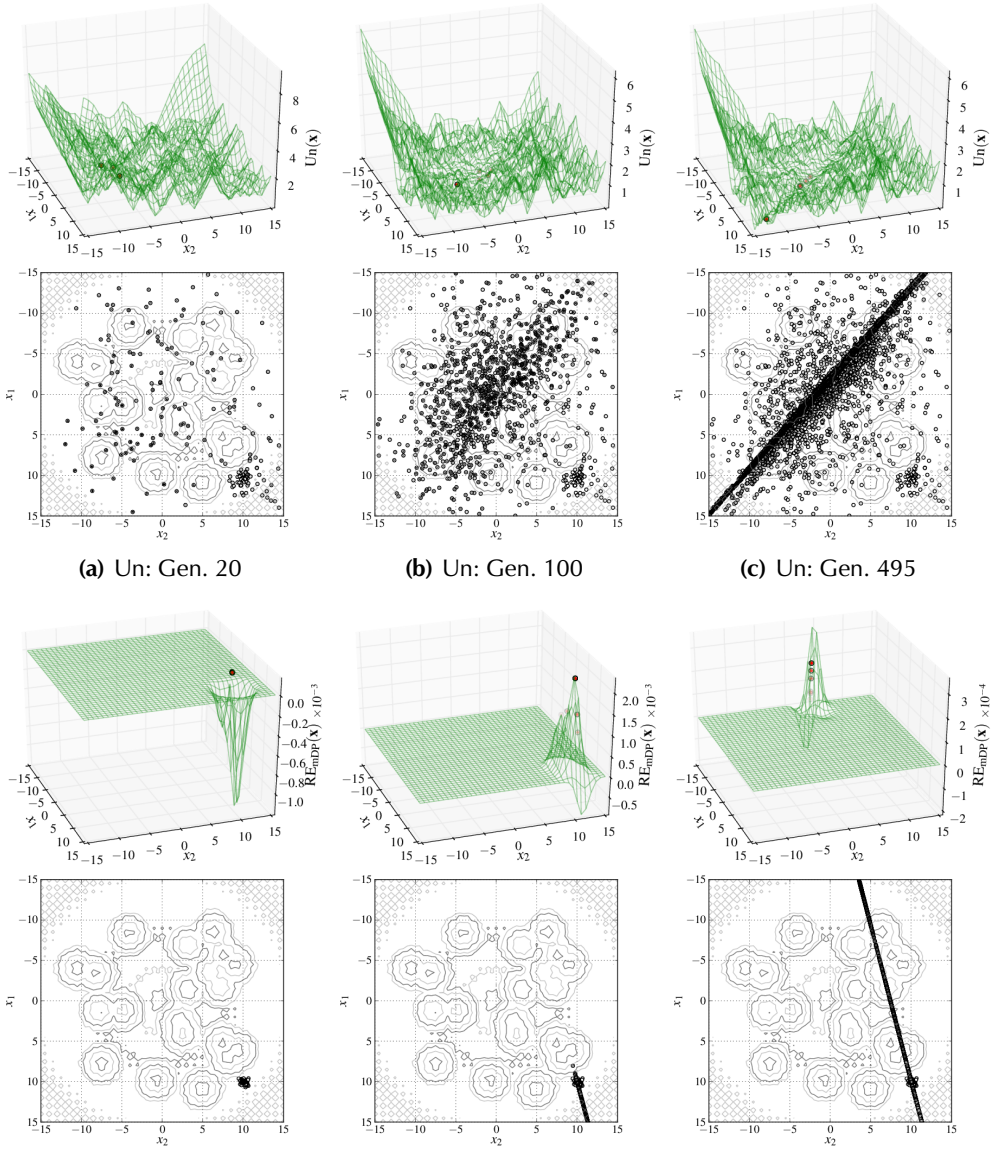


(a) Non-dominated Sorting **(b)** Crowding-distance Criterion

Figure 5.3 Pareto-based Selection Illustrated. Non-dominated sorting in (a) assigns each candidate solution an index representing the front to which it belongs: The first front is non-dominated, and the later fronts are non-dominated if all lower-indexed points are removed. To discriminate between solutions on the same front, the crowding distance criterion in (b) first of all retains the extremal points, and then determines a ranking for the remaining points based on maximizing the distance between their direct neighbors on each side. Figures courtesy of [**Emmerich et al., 2006**].

Different authors have used this crowding-distance Pareto-based selection scheme for selecting on quality and exploration value in parallel. [**Mouret, 2011**] and [**Lehman et al., 2013**] use it to select on quality and distance-based novelty (*sparseness*), while in [**Graening et al., 2010**, **Reehuis et al., 2011**] it is used to select on quality and learning-based novelty (prediction error and dispersion in predictions). [**Schaul et al., 2011**] apply Pareto-based selection, but instead of using crowding distance, they randomly pick a single point from the first front. Furthermore, they use Pareto-based selection on expected improvement (ExI, see Section 2.3.1), which they derive from quality predictions and learning-based novelty expressed as predictive variance, in parallel to, *again*, learning-based novelty expressed as predictive variance. This hybrid multiobjective scheme — weighted aggregation and Pareto-based selection — seems unnecessarily complicated, although it might provide some benefit as via ExI, learning-based novelty is not directly involved in the ranking, but scales the distribution around the quality prediction used to calculate ExI.

Being a set-based selection scheme that promotes extremal solutions, using crowding-distance Pareto-based selection shows clear stagnation combined with the $(3,6)$-CMA-ES recommended for two-dimensional search spaces, see Chapter 2; in the experiments, we therefore use a $(6,12)$-CMA-ES. In Figure 5.4, the results of using this selection scheme together with the selected exploration measures Un and



(a) Un: Gen. 20 **(b)** Un: Gen. 100 **(c)** Un: Gen. 495

(d) RE_{mDP}: Gen. 20 **(e)** RE_{mDP}: Gen. 100 **(f)** RE_{mDP}: Gen. 495

Figure 5.4 Pareto-based Selection. The plots show the solutions generated by a single run on test problem AckleyInfeasible, of a (6,12)-CMA-ES selecting on quality, and Un (in (a) through (c)) or RE_{mDP} (in (d) through (f)) using non-dominated sorting, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored). The top-row plots display the development of the Un scores (in (a) through (c)) and RE_{mDP} scores (in (d) through (f)), where the red dots represent the lastly-selected parent population. In both setups, the CMA-ES is unable to target optima and results in sampling solutions from a search-space-wide, line-shaped distribution.

RE_{mDP}, see Section 5.1, are displayed. While being elegant in the sense of omitting weight parameters, the applied Pareto-based selection prevents the CMA-ES from targeting optima and for both exploration measures makes that it samples points from a line-shaped search distribution extending between the edges of the search space.

5.3 · Separate Exploration Phases

While exploring, the search should generally diverge, but in exploiting it should converge. Therefore, instead of dynamically combining quality-optimization and exploration objectives, we follow [**Cuccu et al., 2011a**] and strictly separate exploration from exploitation. The idea is to use a *separate exploration phase* to find a new starting position for the quality-based optimization. In the exploration phase, ranking solutions is done on exploration value, and in the optimization phase, selection is on quality only.

This separation requires switching between phases and thus a way of deciding when to end either the exploitation phase or the exploration phase, expressed as stopping conditions for the search algorithm. Through its convergent nature, for the exploitation phase stopping conditions are available. A list of possible stopping conditions for the CMA-ES optimizing on quality is given in [**Auger and Hansen, 2005**]. We follow [**Cuccu et al., 2011a**] and use a stopping condition based on the stepsize of the search distribution. For the exploration phase, being divergent, it is less straightforward to formulate a stopping condition. An idea is looking at the development of the solutions' quality: After deteriorating, improving quality values can indicate that a "ridge" has been crossed, potentially entering a new optimum's basin of attraction. How exactly to determine the trend in the development of quality values, for instance the size of the time window over which to look, has to be thought of.

For the application at hand, however, involving infeasible regions, we can formulate a stopping condition based on feasibility. Arrival in infeasible space will be used as stopping condition, assuming that this approach is sufficient for leaving basins of attraction of earlier-converged-to optima. Intuitively, one might suggest passing through the infeasible region entirely before switching to quality-based optimization. Yet, for the actual problem in the domain of aerodynamic optimization of flow bodies, it is uncertain whether a feasible region will lie beyond it, see Chapter 6. A variable penalty is therefore included in the quality value based on the amount of infeasibility,

allowing the quality-based search to return to feasible space.

Because of the difficulty in coining a general stopping condition for the exploration phase, [**Cuccu et al., 2011a**] opt to not actually perform an exploration phase but instead select the solution, from all points already visited, that scores best on the used dynamic exploration measure at that moment in time. As in our problem setup we start from a fixed initial solution, instead of an initial population distributed widely over the search space, we *have* to use an *active* exploration phase, as in this scenario all solutions that have been visited up to a certain point can reasonably be stated to be in the basin of attraction of an earlier-converged-to optimum. By active we mean that new, unseen solutions are sampled during exploration. This way, there is the possibility of skipping over high-quality regions, as we do not select on quality, brought to bear by [**Cuccu et al., 2011a**] as well in motivating their passive exploration phase. However, in our approach, laid out below, the regions traversed in the exploration phases are not prevented to be visited in a later quality optimization or exploration phase.

Thus, checking feasibility of solutions allows for a convenient stopping condition for the exploration phase. It does, however, require evaluating solutions on quality in order to determine their feasibility. Strictly speaking, the exploration phase can be stated to "waste" quality evaluations, as it does use quality function evaluations but only for checking feasibility, not for driving the search directly.

5.3.1 · Alternating Restart Scheme

As presented in [**Reehuis et al., 2013b, Reehuis et al., 2013c**], we employ a restart scheme that alternates between phases of exploration and quality optimization, and starts with a quality-optimization phase. The scheme is summarized in Technical Note 5.1.

At the start of both the exploration and the quality-optimization phase, the underlying search method has to be reinitialized. A different or differently-configured search algorithm can be used per phase, but we employ the same CMA-ES for both. Reinitializing means for the CMA-ES that all variables involved in shaping its search distribution are reset, including the stepsize which is reinstated to its, problem-specific, initial value σ_{init}.

As stated before, each phase requires a stopping condition to be formulated for the search method. The quality-optimization phase will be stopped when the CMA-ES'

Technical Note 5.1 Alternating Restart Scheme for the CMA-ES

1) **Quality-optimization phase**:

   - Initialize stepsize at σ_{init};
   - Rank solutions on quality value;
   - **Stopping condition**: Stepsize smaller than $0.5 \times \sigma_{\text{init}}$.

   Take the best-ranked solution of the current generation and

   ⋆ add it to the archive of found optima;

   ⋆ start the exploration phase from this solution.

2) **Exploration phase**:

   - Initialize stepsize at σ_{init};
   - Rank solutions on exploration value, using
     - Un, reference set: All solutions generated in quality optimization;
     - RE_{mDP}, reference set: All solutions evaluated on quality.
   - **Stopping condition**: Stepsize smaller than $0.5 \times \sigma_{\text{init}}$, or the best-ranked solution of the current generation
     1. exceeds a *hard*-limit quality value;
     2. exceeds a *soft*-limit quality value for a number of generations;
     3. violates other domain-specific feasibility conditions.

   Start the quality-optimization phase from the solution

   ⋆ with the highest-found Un score in the current exploration phase, or

   ⋆ best-ranked on RE_{mDP} in the current generation.

Quit the restart scheme when all available quality evaluations have been used, at any point during execution.

stepsize becomes smaller than $0.5 \times \sigma_{\text{init}}$, considering the search to have converged to an optimum. The exploration phase is also stopped when the stepsize is smaller than $0.5 \times \sigma_{\text{init}}$, to account for possible stagnation, and is stopped when infeasible space is entered, considered such when the best-ranked solution with respect to the exploration measure:

1. Exceeds a *hard*-limit quality value — for AckleyInfeasible, higher than 30;
2. Exceeds a *soft*-limit quality value for a number of consecutive generations — for AckleyInfeasible, higher than 22 for 5 consecutive generations;
3. Violates other domain-specific feasibility conditions — these are not present in AckleyInfeasible.

When applying Un in the exploration phases, search is performed on a *static* landscape as the used reference set consists of the solutions generated during the quality-optimization phases. The solutions that are selected in the exploration phase, i.e., the parent individuals, are evaluated on quality for feasibility checking, but not added to the reference set. Using Un, we aim to put maximum distance to solutions in the regions of attraction of the converged-to optima.

When applying RE_{mDP} in the exploration phases, search is performed on a *dynamic* landscape: The reference set for training the model contains all solutions evaluated on quality, so including the solutions that were evaluated in the exploration phases for feasibility checking. This is required because the maximum of the interestingness landscape is close to the lastly-sampled solutions. We have to keep adding solutions to the reference set to make this maximum and thereby the search move away from where exploration was started. Per generation, the generated offspring serve as test set for determining the modeling errors using DP, see Section 4.3.

The alternating restart scheme is only allowed to switch between phases if in the last phase, exploration and quality-optimization phases alike, at least one feasible solution was generated with respect to conditions 1. and 3. listed above. If not, instead of switching phases, the last type of phase is started again, from the next *restart point in line*. For an exploration phase, this means that not the best-ranked solution from the last generation of the previous quality-optimization phase is taken, but the runner-up solution, continuing in that fashion and ending-up in a before-last generation if need be, after multiple attempted reinitializations. For a quality-optimization phase

this means, in case of Un, that the runner-up solution of the previous exploration phase is used, and so forth, and, in case of $\mathrm{RE}_{\mathrm{mDP}}$, that the runner-up solution in the last generation of the previous exploration phase is used, moving back into earlier generations if necessary, like described above.

Using separate exploration and exploitation phases omits the need to perform a clustering step to isolate found optima: The best solution of the last generation of a quality-optimization phase is stored as approximated optimum. Through the exploration phases, the distance between the found optima, with respect to the used domain-specific distance measure, is promoted, either directly in Un, or implicitly in $\mathrm{RE}_{\mathrm{mDP}}$ by aiming for maximum learning progress.

Level-set Approximation

A level set contains all solutions in the search space with quality equal to or better than a given threshold quality value, see Section 3.2.1. Level-set approximation aims to represent the level set using a limited number of solutions, which are therefore to be spread optimally over the area in which solutions satisfying the level-set constraint lie. The distance between solutions is determined using the generally-applicable Euclidean distance between the solution vectors, or a different domain-specific distance measure. A level-set approximation is valued by the diversity in the set of solutions that make up the approximation, which is aimed to be maximal according to the used set diversity measure. This set diversity measure applies the domain-specific distance measure to pairs of solutions in the set.

One successive exploration and exploitation phase can combined be seen as a variation operator in approximating a level set, delivering a found optimum as a new solution that is to be considered for the approximation. Such a variation operator would be applicable in a scheme such as *ELSA* [**Emmerich et al., 2012, Emmerich et al., 2013**] that aims to approximate a level set with a given threshold f_{thres} using a fixed-size population. However, considering the computational cost involved in applying this "variation operator", one might want to keep all generated candidate solutions and in a post-processing step determine to adopt a certain f_{thres} based on the solutions generated.

This is exactly what is done in running the alternating restart scheme and afterwards determining the diversity of all possible level sets in its result set, i.e., the set of found optima, as laid out in Section 3.2.1. That way, no pre-defined f_{thres} is used,

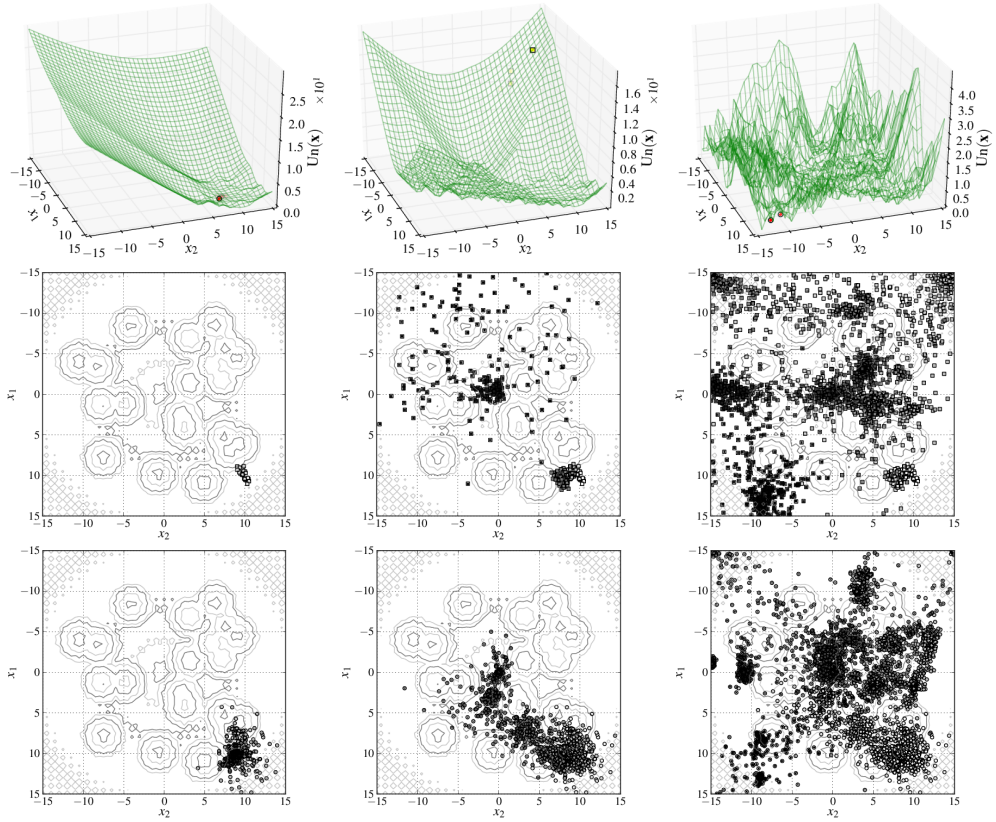
and although one does not optimize on diversity, the distance between found optima is promoted by exploring for restart points that lie far away from earlier-generated solutions. Nevertheless, should one have a pre-defined f_{thres} in mind, it can be used as additional criterion for stopping the quality-optimization phases. In the current setup, governed by a minimal stepsize to be reached, a quality-optimization phase will only stop when a solution has been converged to that has a sufficient level of local optimality. Technically, however, within a level-set approximation, local optimality of the constituents is not a requirement, only satisfaction of the level-set constraint.

Experiments

We apply the alternating restart scheme on test function AckleyInfeasible, running on top of a CMA-ES, with settings as described in Section 5.1. Within the exploration phases, we test Un, as defined and with settings described in Section 4.2.1.2, and RE_{mDP}, as defined and with settings described in Section 4.2.2.3 (training data of $\gamma = 5$ successive generations of evaluated solutions per local model stack). A third, simplified approach is included, *line explore*, inspired by the dynamics of the RE_{mDP} exploration phases. It features a similar division in quality-optimization and exploration phases, but exploration is performed by just sampling a random vector from a multivariate normal distribution, $\mathcal{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}_n\right)$ with $\sigma = 0.3$ (equal to σ_{init} for AckleyInfeasible), and iteratively adding it to the start solution, resulting in moving in a straight line through the search space. After each addition, the resulting solution is evaluated on quality to determine whether one of the infeasibility stopping conditions was met.

Plots of all solutions generated in a single run by each of the three methods, the alternating restart scheme using Un, the alternating restart scheme using RE_{mDP}, and the alternating restart scheme using line explore, are shown in Figure 5.5 through 5.7. Of these three single runs, the results are summarized in Figure 5.8 using the quality/diversity-Pareto interpretation described in Section 3.2.1. Furthermore, quantifying the surface dominated in quality/diversity space as a hypervolume score, the results of 100 runs per method are compared in Figure 5.9.

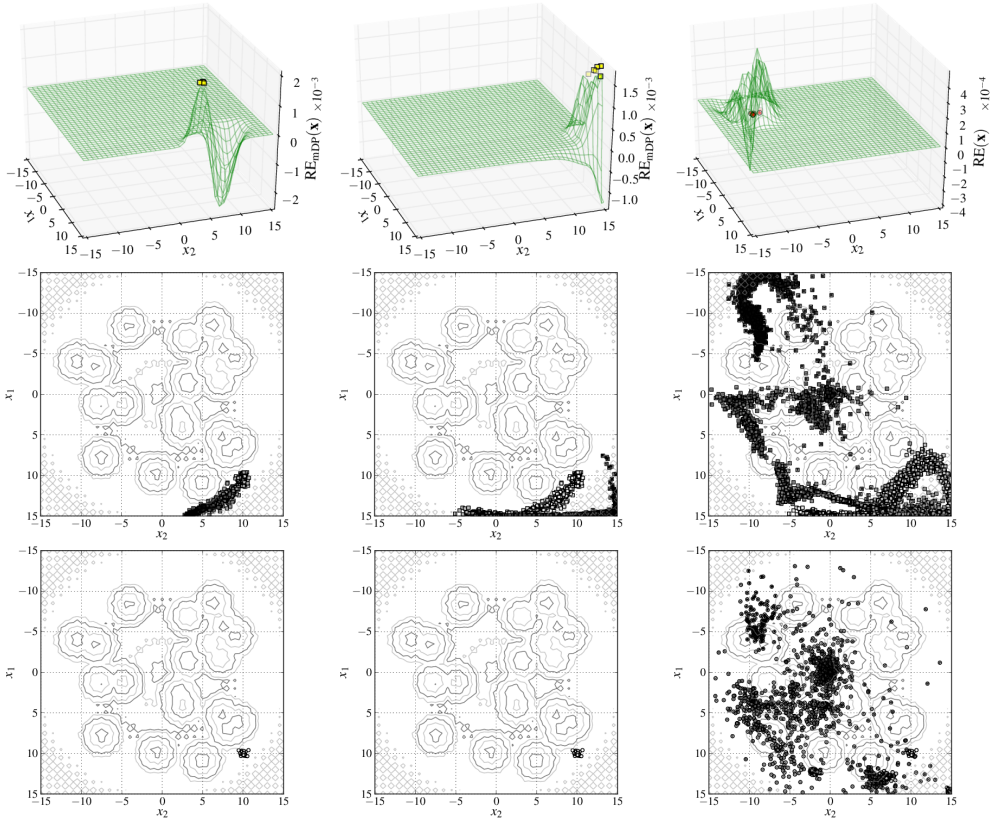
Looking at Figure 5.5, after quick convergence to a local optimum close to the initial solution of $(10, 10)$ (equal to the behavior of the unassisted CMA-ES, see Figure 5.2), an exploration phase is started in the direction of the center of the search space, where the Un landscape shows higher values. With exploration quickly ending up in



(a) Generation 40 **(b)** Generation 200 **(c)** Generation 890

Figure 5.5 Restarting — Un. The solutions generated by a single run on test problem AckleyInfeasible, of the restart scheme based on a (6,12)-CMA-ES alternating between optimization on quality and exploration using uniqueness, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored). The middle-row plots show the solutions generated in the exploration phases and the bottom-row plots the solutions generated in the quality-optimization phases. The top-row plots display the development of the Un scores, where the red dots represent the lastly-selected parent population in case of quality optimization and the yellow squares in case of exploration. The alternating restart scheme is able to leave the lower-right corner of the search space, and is able to locate the global optimum at (0,0), as well as multiple alternative optima.

infeasible space, a few successive exploration phases are required for exploitation to leave the lower-right corner of the search space. In generation 122, exploitation is restarted in point (9.7,5.2), after which it is able to autonomously locate the global optimum. The alternating restart scheme continues, finally delivering a result set

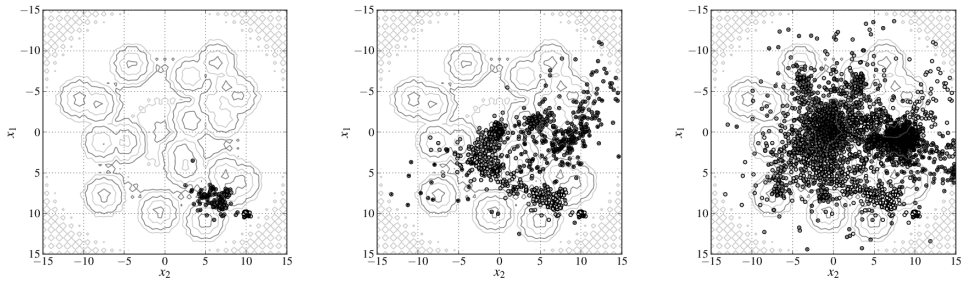


(a) Generation 40 **(b)** Generation 200 **(c)** Generation 805

Figure 5.6 Restarting — RE_{mDP}. The solutions generated by a single run on test problem AckleyInfeasible, of the restart scheme based on a (6,12)-CMA-ES alternating between optimization on quality and exploration using RE_{mDP}, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored). The middle-row plots show the solutions generated in the exploration phases and the bottom-row plots the solutions generated in the quality-optimization phases. The top-row plots display the development of the RE_{mDP} scores, where the red dots represent the lastly-selected parent population in case of quality optimization and the yellow squares in case of exploration. Exploration using RE_{mDP} constrains the width of the search distribution, showing relatively-slow but directed movement through the search space, and allowing multiple optima, including the global optimum at (0,0), to be located.

containing 20 (local) optima in using 6000 quality evaluations.

Whereas exploration on Un increases the width of its search distribution after being initialized, exploration on RE_{mDP}, see Figure 5.6, shows the familiar interestingness peak that moves through the search space and that keeps the width of the search



(a) Generation 20 **(b)** Generation 100 **(c)** Generation 470

Figure 5.7 Restarting — Line Explore. The solutions generated by a single run on test problem AckleyInfeasible, of the restart scheme based on a (6,12)-CMA-ES alternating between optimization on quality and line exploring, at three time steps (recently-generated solutions are dark-colored, early-generated solutions are bright-colored). Following a straight line for finding restart points for the exploitation phases is a relatively simple approach that allows for locating multiple optima, including the global optimum at (0,0).

distribution at moderate levels. From the plots of generation 805, it is clear to see that exploration brings quality-optimization to new areas to exploit. In doing so, it moves through the search space at a lower speed than exploration on Un, and as such delivers a result set containing fewer optima, 8, after using 6000 quality evaluations.

Different runs of the alternating restart scheme reach a different number of generations. This is because the exploration phases do not require evaluating all solutions but only the selected parents, thereby allowing for more generations in total. For the restart scheme using line explore, see Figure 5.7, the exploration phases are not counted in the number of generations, as only one solution is generated and evaluated per iteration. As can be seen, line explore is a simple approach that allows the CMA-ES to leave the region of attraction of the optimum found by default, to locate the global optimum, and produces 15 alternative optima in using 6000 quality evaluations.

From Figure 5.8, it is visible that with 20 approximated optima, Un generated more points than the other methods, resulting in the highest hypervolume score of 63.2. Line explore delivered only a few optima less, but the majority of these were approximations of the global optimum (11 of 15 found in total). Similar solutions do not contribute much to the surface dominated in quality/diversity space, and hence its hypervolume score of 37.8 is the lowest of the three runs. RE_{mDP}, with its slower-moving exploratory behavior, delivers only 8 approximations of optima, but 6 of these are sufficiently distinct to result in a hypervolume score of 39.6.

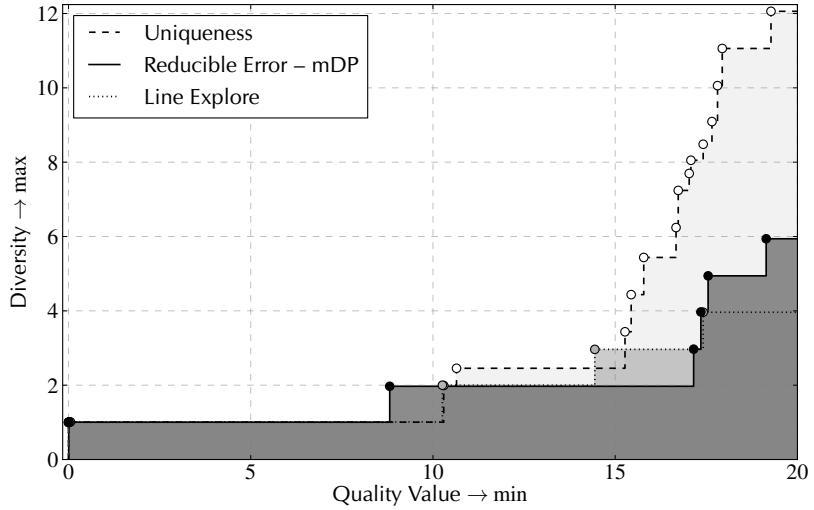
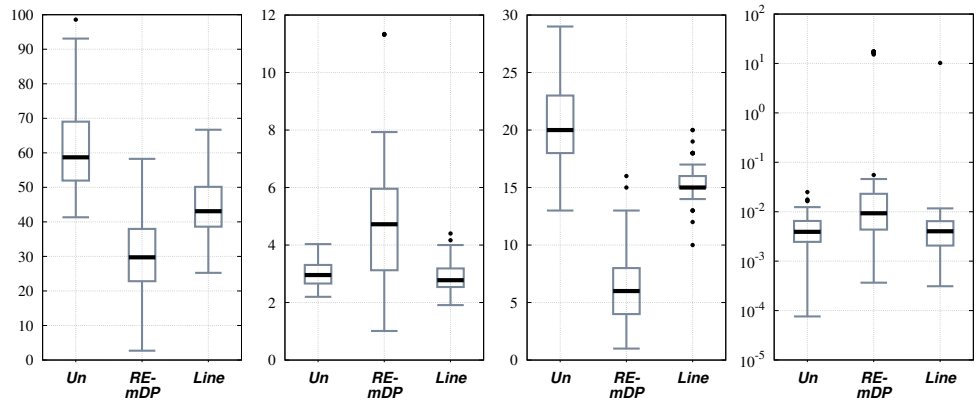


Figure 5.8 Single-run Performance Comparison. Per single run on AckleyInfeasible, the quality threshold and diversity score is plotted for each possible level set in the result set. Interpreting these pairs jointly as a Pareto-front approximation, we visualize the dominated surface in quality/diversity space. Based on a reference point of (20,0), uniqueness has an obtained hypervolume of 63.2 (result set of 20 solutions), reducible error of 39.6 (result set of 8 solutions), and line explore of 37.8 (result set of 15 solutions).

Finally, based on statistical comparison of attained hypervolume scores, see Figure 5.9, Un clearly outperforms RE_{mDP}, while line explore ends up in between. Nonetheless, looking at hypervolume scores only does not show the complete picture. The scores are greatly influenced by the number of solutions in a result set, through the used set diversity measure D_{SP} that maximally returns a score equal to this number (see Section 3.2.1), and through the fact that larger sets are likely to dominate a larger surface in quality/diversity space. We therefore include a statistical comparison of the number of solutions found per method, showing a similar spread over the three methods as for the hypervolume scores. When we then, per run, divide the hypervolume score by the number of solutions found, Un and line explore show similar hypervolume per solution, but RE_{mDP} now has a higher median outcome. It should be noted here that the RE_{mDP} hypervolume-per-solution scores show much more variation though, in positive and negative directions. Lastly, another performance aspect considered is the quality value of the best solution found, which is to be minimized. Here Un and line explore again show similar performance, with RE_{mDP} falling a bit behind.



(a) Hypervolume **(b)** Hypervol. per Sol. **(c)** Num. Solutions **(d)** Best Quality Found

Figure 5.9 Statistical Comparison. Per method, of 100 runs on test function AckleyInfeasible, the hypervolume score of each result set in quality/diversity space is calculated, displayed in (a). The hypervolume score is influenced by the number of solutions per result set, displayed in (c). Hypervolume score divided by the number of solutions in the result set is therefore displayed in (b). Lastly, another performance aspect considered is the to-be-minimized quality value of the best solution found per run, shown in (d). Un produces the largest result sets and this helps to attain the greatest hypervolume scores of the three methods. When hypervolume is expressed per solution, it is clear that despite its lower total hypervolume scores, RE_{mDP} is still a method to be considered for the actual real-world application.

5.4 · Summary

When integrating dynamic exploration criteria into quality-based search, one has the choice between selecting on quality and exploration criterion in parallel, or dividing the search into distinct exploitation and exploration phases. Although the first approach is easier to implement, not requiring conditions for switching phases and performing reinitialization at the start of a new phase, it is noted that exploitation and exploration induce inherently conflicting dynamics. Increasingly local search is performed in exploiting, while exploration continuously drives the search away from areas that were visited already. Separation in dedicated phases therefore constitutes the more efficient approach, preventing that the objectives can hinder each other.

A certain level of to-be-attained accuracy, expressed as the stepsize of the search distribution, can intuitively be used as stopping for the exploitation phase. On the other hand, because of its divergent nature, formulating a general stopping condition for the exploration phase is less straightforward. The real-world application to which the developed methods are to be applied, however, includes regions with intolerable

solutions in its search space. After converging in the exploitation phase, we can start exploring until we end up in such an infeasible region, under the assumption that this puts enough distance to the region of attraction of the earlier converged-to optimum. Quality-based search is then restarted from this point close to the boundary of infeasible space, including a variable penalty to be able to move back to a feasible solution.

This results in what is termed the alternating restart scheme, here, but not specifically, run on top of the *CMA-ES*, and tested in employing the two promising exploration measures resulting from Chapter 4: Distance-based novelty measure *uniqueness* (Un) and interestingness measure *reducible error*, based on memories of *dispersion-in-prediction* novelty (RE_{mDP}). Furthermore, a third baseline method of exploring in a random straight line was included.

On the two-dimensional instance of test problem *AckleyInfeasible*, Un performs best with respect to hypervolume dominated in quality/diversity space, an approach laid out in Chapter 3 for measuring performance with respect to the aim of finding diverse high-quality solutions. RE_{mDP} comes in last, and also has the smallest number of solutions found per run. When accounting for this, RE_{mDP} does show the highest hypervolume-per-solution scores.

RE_{mDP}-induced exploration is slow in moving through the search space compared to the other two methods. This results in smaller result sets, and lower hypervolume scores. The actual application in optimization of airfoils that we have in mind for the developed methods involves a higher-dimensional search space. Moreover, Ackley-Infeasible might not sufficiently capture the essence of this application that includes more infeasibility characteristics. In the following chapter, we will therefore employ the alternating restart scheme with each of the three exploration approaches on the actual real-world problem.

6

Application to Airfoil Optimization

In design optimization of airfoils, one is typically faced with high-dimensional search spaces that are constrained by *geometric* and *manufacturing* requirements. In certain problem instances, the simulator used for modeling the aerodynamic properties of candidate geometries will only return trustworthy results if the geometry under consideration is located well in *feasible* space, that is, not violating any such constraint. The search thus needs to be started from a basic solution known to be feasible. Furthermore, the *mesh grid*, i.e., the discretization of the space around a geometry, used in simulating, will be distributed according to this basic geometry. Therefore, at each stage of the search, the candidate solutions have to resemble the basic shape to a critical extent for the simulator to produce meaningful results, or produce results at all.

When trying to find *innovative solutions* in an automated way, we are hindered by the fact that the search needs to be started from basic feasible solutions. Though applying a stochastic optimization method, multiple instances of the search may converge to largely-similar optimized solutions. In order to uncover innovative solutions, we have to escape the basin of attraction of the solution that optimization typically converges to when started from a certain basic solution. Moving sufficiently far away, guided by some exploration scheme, has to be done stepwise, as we need to check that we are still in or close enough to feasible space to be able to start quality-based optimization again.

In Chapter 3, we motivated that a tangible approach for finding innovative solutions

is through aiming for a *diverse* set of *high-quality* solutions. With increased diversity and quality in a set of solutions, the potential of innovative solutions being present amongst these increases as well, ultimately to be determined by a human domain-expert. Diversity is determined with respect to a *domain-specific distance measure*. For the considered two-dimensional aerodynamic optimization of flow bodies, the difference between solutions will not be expressed by looking at the solution vectors in the search space (the *genotypes*), but by comparing other solution representations like the decoded two-dimensional shapes of the designs (the *phenotypes*). Note that instead of solely with these airfoil geometries, *patents* are typically concerned with the flow phenomenon that a certain geometry gives rise to, which is thus another possible source for determining difference between solutions.

This chapter deals with applying the restart scheme that was laid out in Chapter 5, alternating between phases of quality-optimization and exploration, to a real-world airfoil-optimization task, using the three tested exploration variants: Distance-based novelty measure *uniqueness* (Un), interestingness measure *reducible error* ($\mathrm{RE}_{\mathrm{mDP}}$), based on memories of *dispersion-in-prediction* novelty, and the baseline method of exploring in a random straight line. First, the domain-specific distance measure is defined in Section 6.1, which will be used in diversity calculations as well as in distance-based novelty measure Un, after which the airfoil optimization test case is described in Section 6.2. Section 6.3 then lists the settings used for the alternating-restart-scheme variants in the experiments, of which the results are described in Section 6.4. A summary of the chapter is given in Section 6.5.

6.1 · Domain-specific Distance Measure

With respect to the domain of two-dimensional aerodynamic optimization of flow bodies, a distance measure is to be defined that will be used in scoring the diversity of sets of solutions after optimization, as well as for determining distance-based novelty of solutions during optimization. The two dimensions refer to the airfoil representation used in simulating aerodynamic behavior, the actual solution vectors from which these two-dimensional representations are derived are of (much) higher dimensionality. One aspect of the domain-specific distance measure is concisely expressing the diversity of sets of solutions, another is assisting the distance-based Un scheme in moving through the search space with reasonable computational cost. To come to an appropriate distance measure, we have to choose a solution representation first, and then a way

of calculating distance between two instances of that representation.

To lay out the different possible sources for determining the distance between solutions in the considered problem class, we sketch the path from solution vector to quality value. For a solution vector that represents a certain airfoil geometry, the aerodynamic behavior of that geometry is approximated by *computational fluid dynamics* (CFD) simulation. The simulated *flow field* around the geometry is numerically abstracted into a quality value, schematically

$$\underset{\mathbf{x}}{\text{genotype}} \xrightarrow[\textit{NURBS}]{\text{decoding}} \underset{\mathrm{P}(\mathbf{x})}{\text{phenotype}} \xrightarrow[\textit{CFD}]{\text{simulation}} \underset{\substack{\mathrm{B}(\mathbf{x})\\ \textit{flow field}}}{\text{behavior}} \xrightarrow{\text{analysis}} \left.\begin{array}{c}\text{properties}\\ \mathrm{prop}_1(\mathrm{B}(\mathbf{x}))\\ \vdots\\ \mathrm{prop}_k(\mathrm{B}(\mathbf{x}))\end{array}\right\} \xrightarrow[\textit{heuristics}]{\text{aggregation}} \underset{f(\mathbf{x}).}{\text{quality}} \tag{6.1}$$

The *genotype* $\mathbf{x}$, consisting of n input parameters x_i, represents an airfoil geometry that is obtained through interpreting the input parameters as *non-uniform rational B-spline* (NURBS) *control points*, resulting in the *phenotype* of the solution, denoted $\mathrm{P}(\mathbf{x})$. CFD simulation is used to generate a flow field visualization around the phenotype, termed the *behavior* of $\mathbf{x}$, denoted $\mathrm{B}(\mathbf{x})$. From this flow field approximation, scores for predefined properties $\mathrm{prop}_j(\mathrm{B}(\mathbf{x}))$ are determined, which are then aggregated into a single quality value according to some heuristic weight vector $\mathbf{w}$,

$$f(\mathbf{x}) = \sum_j w_j \,\mathrm{prop}_j(\mathrm{B}(\mathbf{x})). \tag{6.2}$$

From the different available characterizations, we determine distance based on the two-dimensional phenotypes, as these allow for an effective but relatively simple domain-specific distance measure. The two phenotypes are sampled equidistantly along the spline, and each sample is again divided into two sets of points, one set representing the *suction* side (side featuring higher air speed, upper side in Figure 6.1(b)) and one set representing the *pressure* side (side featuring lower air speed, lower side in Figure 6.1(b)) per airfoil. The *summed-up Hausdorff distance* [**Olhofer et al., 2002**] is then applied on the suction sets and the pressure sets. It is defined as

$$\mathrm{d}_{\Sigma\mathrm{H}}(A,B) = \sum_{\mathbf{a}\in A} \min_{\mathbf{b}\in B} \|\mathbf{a}-\mathbf{b}\| + \sum_{\mathbf{b}\in B} \min_{\mathbf{a}\in A} \|\mathbf{a}-\mathbf{b}\|, \tag{6.3}$$

that is, for each two-dimensional point $\mathbf{a} = (a_1, a_2) \in A$, calculate the distance to the

closest point $\mathbf{b} \in B$, and vice versa, and sum all these distances. To determine the distance between two solutions $\mathbf{x}_1$ and $\mathbf{x}_2$, we add the summed-up Hausdorff distance between the suction side samples and the distance between the pressure side samples,

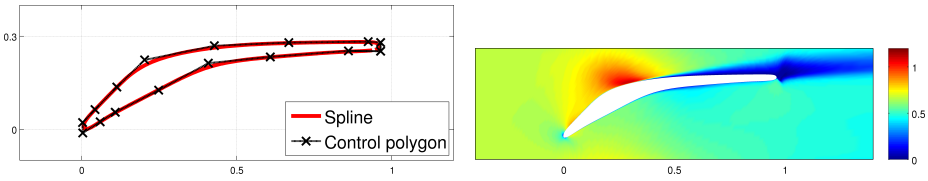
$$\begin{aligned} d_{\text{airfoil}}(\mathbf{x}_1, \mathbf{x}_2) \; &= d_{\Sigma H}(\text{suction}(P(\mathbf{x}_1)),\, \text{suction}(P(\mathbf{x}_2))) \quad + \\ &\quad d_{\Sigma H}(\text{pressure}(P(\mathbf{x}_1)),\, \text{pressure}(P(\mathbf{x}_2))) \,. \end{aligned} \tag{6.4}$$

In this formulation of domain-specific distance, the fact that certain differences are generally weighted as more important by domain experts is omitted, for instance characteristics of the *trailing edge*, see Figure 6.1, that can induce important features in the resulting flow field. Structural weighting could be included in the way that domain-specific distance is calculated, although it is likely to prove difficult to comprehensively account for all aspects that a human engineer might consider.

The most-characterizing representation is the aerodynamic behavior that a geometry gives rise to, containing relevant information available through simulation, reflected in the fact that quality is derived from it as well. While more involved, distance could be calculated between *Mach number distributions*, see Figure 6.1, for instance by transforming an airfoil geometry into a comparison geometry (*morphing*), and having the surrounding flow field representation morph with it. Pixel-wise, difference can then be determined between the resulting morphed image and the comparison image. Alternatively, distance is determined by comparing *Mach-number-distribution profiles* along the airfoils, much like the way distance is determined between the phenotypes by using summed-up Hausdorff distance. At the end of this chapter, examples of Mach-number-distribution profiles are shown (see Figure 6.9, top row). In using this abstraction, however, we might not capture certain relevant features that are taken into account in calculating distance over the full Mach number distribution.

6.2 · Test Case: Two-dimensional Stator-blade Optimization

Optimization is performed on the *outlet guide vanes* (OGVs) of the *low-pressure compressor* in a small *turbofan* engine that is intended for propelling small business jet aircraft. Two-dimensional airfoil representations are used, involving 16 control points and thus 32 decision variables, see Figure 6.1. OGVs are *stator blades*, non-moving blades intended for straightening the air flow, arranged around a central axis in a cylindrical fashion. The compressor, displayed as a holistic unit in Figure 6.2, is in this case divided in a low-pressure compressor followed by a high-pressure compressor.



(a) Spline Representation **(b)** Mach Number Distribution

Figure 6.1 Initial Geometry. In (a), the 16 *control points* and their *non-uniform rational B-spline* (NURBS) decoded geometry is shown. In (b), the *Mach number distribution* (the relative air speed divided by the local speed of sound) of the simulated *flow field* around the geometry is displayed, showing a deviating outflow of $-5.96°$ for the initial geometry. The control points can move on the horizontal and vertical axis, in all directions, as before simulation the geometry is translated to have its *leading edge* (left-most point) in (0,0), and is normalized by the *chord length* (distance between the leading edge and the *trailing edge*, right-most point).

After each compressor stage, stator blades are included to straighten the outflow.

The test case is a problem instance in which optimization needs to be started from a basic initial solution, displayed in Figure 6.1, as the search space features large infeasible regions containing solutions for which the quality cannot be determined through simulation. Simulating aerodynamic behavior is done using an in-house, quasi-three-dimensional *Navier–Stokes flow solver*, *Honda Software for Turbomachinery Aerodynamics Research* (HSTAR) [**Arima et al., 1999**]. Navier–Stokes equations are partial differential equations, involving multivariate functions, that express the motion of *fluids*. HSTAR is a deterministic simulator, hence simulation is not prone to noise, i.e., varying outcome of repeated simulation of a single solution [**Kruisselbrink, 2012**]. However, in approaching infeasible space, non-stationary uncertainty, i.e., locally-increased irregularity in quality values [**Kruisselbrink, 2012**], will be observed as the simulator has increased difficulty in reliably approximating the flow field around the candidate geometries.

6.2.1 · Quality Function

The fitness of a solution is based on seven properties that are aggregated into a *to-be-minimized* quality value, namely

- the resulting deviation angle of the air flow, that is, the outflow angle α_2 minus the target outflow of $0°$ (minus tolerance of $0.3°$),
- the coefficient ω of the occurring pressure loss,

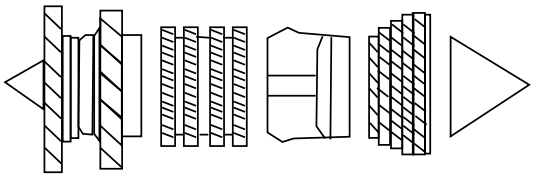


Figure 6.2 Sketch of a Gas Turbine. The *fan* (a) pulls air into the engine. Part of this air is compressed in the *compressor* (b) and then forced into the *combustion chamber* (c), where it is mixed with fuel and ignited. The resulting hot, high-energy gases go into the *turbine* (d), causing the turbine blades to rotate, and the turbine to drive the compressor. The *nozzle* (e) is the exhaust duct of the engine. Image and description courtesy of [**Hasenjäger et al., 2005**].

- the extent of violation of four minimal-thickness constraints, i.e., a listed thickness has to be larger than its respective minimal thickness value, all values expressed as percentage of the blade's chord length cl, see Figure 6.1,
  - leading edge radius r_{le}, larger than $0.5\% \times \text{cl}$,
  - trailing edge radius r_{te}, larger than $0.5\% \times \text{cl}$,
  - maximum thickness $\theta_{\max}$ of the blade, orthogonal to the outflow angle, larger than $5\% \times \text{cl}$,
  - minimum thickness $\theta_{\min}$ of the blade, orthogonal to the outflow angle, larger than $0.5\% \times \text{cl}$,
- and a penalty depending on the extent to which the simulator has not converged in solving the system of flow equations; $\varepsilon_{\text{solver}}$ is the observed residual value, and the maximally allowed residual is 10^{-4}.

The to-be-minimized quality function is defined as

$$\begin{aligned} f(\mathbf{x}) \;=\; & w_{\alpha_2} \cdot g\left(|\alpha_2(\mathbf{x}) - 0°| - 0.3°\right)^2 \;+ \\ & w_{\omega} \cdot \omega(\mathbf{x})^{1.5} \;+ \\ & w_{\text{th}} \cdot \Big(g\left(0.5\% \times \text{cl} - r_{\text{le}}(\mathbf{x})\right)^2 + g(0.5\% \times \text{cl} - r_{\text{te}}(\mathbf{x}))^2 + \\ & \qquad g(5\% \times \text{cl} - \theta_{\max}(\mathbf{x}))^2 + g(0.5\% \times \text{cl} - \theta_{\min}(\mathbf{x}))^2 \Big) \;+ \\ & w_{\varepsilon} \cdot \left\lceil g\left(\varepsilon_{\text{solver}}(\mathbf{x}) - 10^{-4}\right) \right\rceil \cdot \varepsilon_{\text{solver}}(\mathbf{x}) / 10^{-4}, \end{aligned} \tag{6.5}$$

with

$$w_{\alpha_2} = 1,\ w_{\omega} = 20,\ w_{\text{th}} = 5 \times 10^6,\ w_{\varepsilon} = 10^5,$$

and

$$g(a) = \max(0, a). \tag{6.6}$$

6.3 · Modus Operandi

The alternating restart scheme, defined in Section 5.3.1, is run on top of a (12,24)-CMA-ES for 5000 generations (120k quality evaluations by the simulator). For search dimension 32, see Section 6.2, the minimally prescribed population sizes for the CMA-ES are (7,14), see Chapter 2. Larger population sizes where chosen to make the search less locally-oriented, in both quality optimization and exploration. The alternating restart scheme is compared against 5 standard optimization runs using the same (12,24)-CMA-ES, running for 1000 generations each (24k evaluations). A problem-specific initial stepsize σ_{init} of 1×10^{-3} is used.

Within the alternating restart scheme, we stop the quality-optimization phase when the stepsize is smaller than $0.5 \times \sigma_{\text{init}}$. The exploration phase is also stopped when the stepsize is smaller than $0.5 \times \sigma_{\text{init}}$, to account for possible stagnation, and is stopped when infeasible space is entered, considered such when the best-ranked solution with respect to the exploration measure

- is in a maximally erratic area for still allowing quality-based search to find a way back to a high-performing solution, i.e.,
    1. exceeds problem-specific hard-limit quality value 1×10^6;
    2. exceeds problem-specific soft-limit quality value 1×10^3, for 5 consecutive generations;
- is strictly infeasible, in which case a penalty is also added to the quality value, i.e.,
    3. the simulator does not converge for it: A variable penalty depending on simulator non-convergence is included, see Equation 6.5;
    4. contains a loop, that is, the lines defining the suction side and pressure side cross, i.e., upper and lower side in Figure 6.1(b): The solution gets the *death penalty*, i.e., its quality value is set to $\gg 1 \times 10^6$;

5. the leading edge is not the lowest point of the blade, see Figure 6.1: The solution gets the death penalty.

The penalty values are included in quality scores of strictly infeasible solutions to assist the search in subsequently leaving infeasible space, in the quality-optimization phase that follows.

In the exploration phases, for escaping the basin of attraction of the lastly converged-to optimum, deviant geometries will be generated for which we will not get an accurate result from the simulator. As the mesh grid used in simulating is prepared for shapes with "standard" characteristics, more resemblant of the basic initial solution, the mesh might not be dense enough around certain areas of the shapes evolved in exploration to get a good outcome. The accuracy of the simulated aerodynamic behavior, however, is not a primary consideration, as long as, roughly speaking, on the path back to feasible space, solutions get increasingly better quality values. This is either through the intrinsic quality values that, despite their initial inaccuracy, should steadily decrease towards feasible space, or through the levels of non-convergence that also decrease in moving towards shapes sufficiently similar to the basic initial solution.

The conditions that lead to the death penalty, on the other hand, are treated differently, as these solutions should be strictly avoided: A loop or the leading edge no longer being the lowest point in a geometry is difficult to eliminate again through optimizing on quality. For this reason, in exploring, all offspring individuals are evaluated to check their feasibility, not only the selected parents, and therefore, generations in the exploration and quality-optimization phases use the same amount of quality evaluations. Moreover, because these forms of infeasibility are difficult to optimize out and because of the resulting costly exploration phases with respect to quality evaluations, we do not try to traverse infeasible space but instead halt exploration when it is entered.

Three variants of exploration are applied with the alternating restart scheme: Distance-based novelty measure uniqueness (Un), interestingness measure reducible error ($\mathrm{RE}_{\mathrm{mDP}}$), based on memories of dispersion-in-prediction novelty , and a baseline method of exploring in a random straight line (line explore). The exploration steps in line exploring, using a multivariate normal distribution with stepsize 0.01 (10 times σ_{init}), are not counted as a generation, as they involve evaluating only a single solution

on quality. Line explore therefore runs for (slightly) less than 5000 generations.

In calculating novelty as DP for RE_{mDP}, as described in Chapter 4, local model stacks consisting of 10 *feed-forward neural networks* (FFNNs, see Chapter 2) are used. In each generation, a new stack is trained on the last $\gamma = 3$ generations of evaluated solutions, using maximally 72 training points per local model stack (cf. the last 5 generations in Chapter 4 and 5, maximally 30 and 60 training points respectively). An ensemble of *disjoined* local stacks is maintained by keeping the local model stack of each third generation, indicated by $\gamma = 3$. Sample points that are infeasible, either through exceeding quality limits or violating one of the strict feasibility conditions, *are* included in training, as these just have a bad intrinsic or penalized quality value.

The FFNNs have linear outputs and a hidden layer of 25 sigmoidally-activated nodes, a bias node, and are trained using improved Rprop [**Igel and Hüsken, 2003**] for a maximum of 10k epochs from random initial connection weights in $[-0.1, 0.1]$. They are fully connected, including direct connections from input to output nodes. All data is used for training, no validation set is used. The CMA-ES and learning-guided restart scheme have been implemented using the Shark Machine Learning Library v2.3.43[1].

6.4 · Results

For each alternating-restart-scheme variant, 20 runs were performed on the task of two-dimensional stator-blade optimization. Sets of 5 standard-optimization runs are used as comparison, which can be seen as a single run of 120k evaluations that was restarted four times, performing 20 of such sets of 5 runs, thus 100 standard-optimization runs in total. Primary performance criterion is the surface dominated in quality/diversity space, laid out in Section 3.2.1, accounting for the diversity in a result set of solutions as well as the quality of these solutions. Measuring this surface leads to a singular hypervolume score that we can use to perform statistical comparison of the four methods: Un, RE_{mDP}, line explore, and 5 standard-optimization runs.

Per method, we select a run with median hypervolume score with respect to all runs of that method and plot the surface dominated by the solutions in its result set, in Figure 6.3. Only solutions with quality value and associated diversity score strictly better than those of the reference point of $(0.11, 0)$, used in the hypervolume calculations, are considered. The quality values are to be minimized: Beyond this quality value of 0.11, solutions are regarded not to be of high-quality. Remarkably,

[1] http://shark-project.sourceforge.net

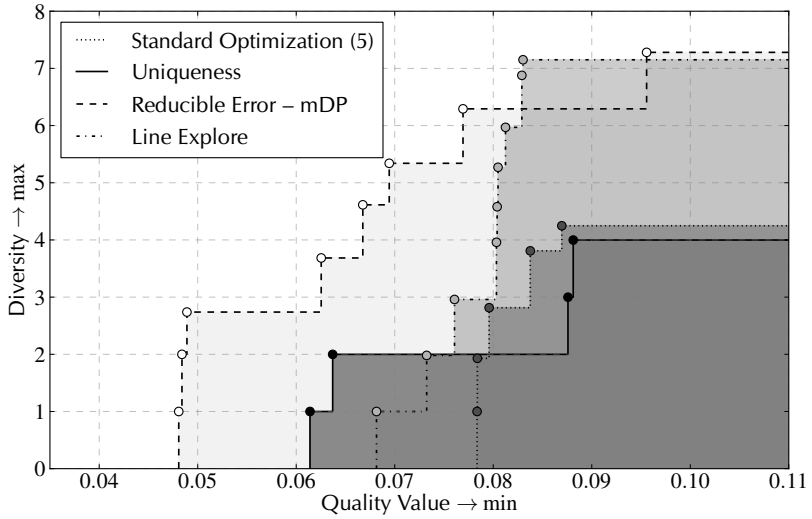
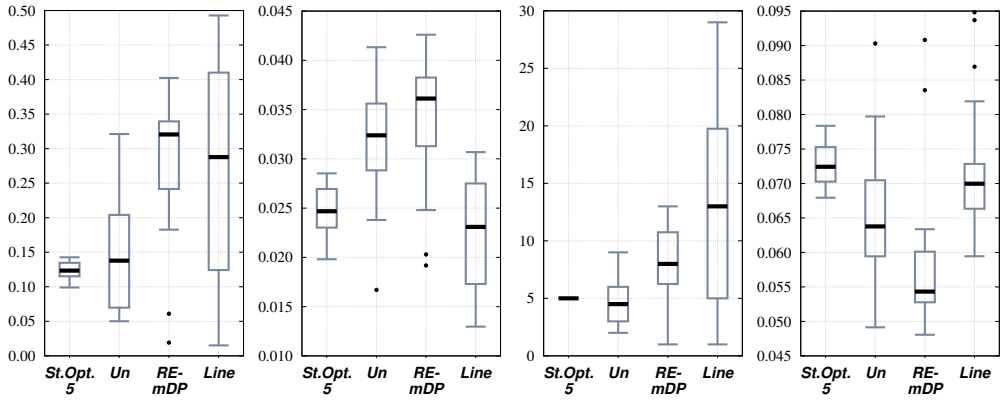


Figure 6.3 Median-run Performance Comparison. Per method, one run with median hypervolume score is selected. Of each median run, the quality threshold and diversity score is plotted for each possible level set in its result set. Interpreting these pairs jointly as a Pareto-front approximation, we visualize the dominated surface in quality/diversity space. Based on a reference point of (0.11,0), standard optimization has an obtained hypervolume of 0.124 (result set of 5 solutions), uniqueness of 0.139 (result set of 4 solutions), reducible error of 0.329 (result set of 8 solutions), and line explore of 0.232 (result set of 9 solutions).

RE_{mDP}, showing worst hypervolume scores on AckleyInfeasible, see Section 5.3.1, is the best performing method on the real-world airfoil-optimization task.

When we further analyze the hypervolume scores, see Figure 6.4, it becomes clear that, like on AckleyInfeasible, line explore does well because of the large result sets it produces. Accounting for this, line explore gets the worst of the hypervolume-per-solution scores. Most notably, RE_{mDP} has, next to the best median hypervolume score, the best median hypervolume-per-solution score, and additionally, it finds the best-quality solutions. Un follows closely in hypervolume-per-solution scores, but through producing smaller result sets, falls behind in hypervolume scores. Nevertheless, next to their good results, both RE_{mDP} and Un include very weak runs with respect to the number of solutions found *and* hypervolume per solution. In these runs, the search ends up in an area of the search space in which no high-quality solutions are found, repeatedly converging to solutions with quality values worse than 0.11, which are therefore omitted in the results.

To explain the difference in performance between RE_{mDP} and Un, we analyze the

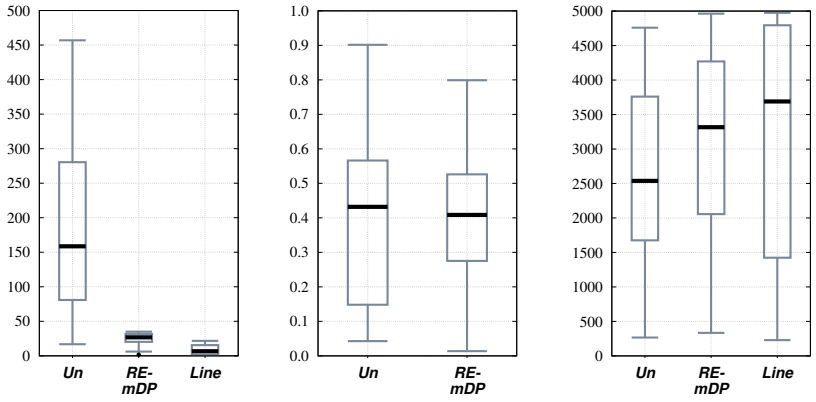


(a) Hypervolume **(b)** Hypervol. per Sol. **(c)** Num. Solutions **(d)** Best Quality Found

Figure 6.4 Statistical Comparison. Per method, of 20 runs performed, the hypervolume score of each result set in quality/diversity space is calculated, displayed in (a). The hypervolume score is influenced by the number of solutions per result set, displayed in (c). Hypervolume score divided by the number of solutions in the result set is therefore displayed in (b). Lastly, another performance aspect considered is the to-be-minimized quality value of the best solution found per run, shown in (d). RE_{mDP} has the best median hypervolume score, the best hypervolume-per-solution scores, *and* finds the best-quality solutions.

alternating-restart-scheme variants in Figure 6.5. Averaged per run, Un needs many more steps (i.e., generations) per exploration phase to cover a similar distance in the decision space, i.e., Euclidean distance between solution vectors, as RE_{mDP}, that is, RE_{mDP} finds the boundary of infeasible space quicker. In contrast to the results on test function AckleyInfeasible, see Section 5.3.1, in the higher-dimensional search space of the stator-blade optimization, selecting on RE_{mDP} induces *faster* exploratory behavior with a larger stepsize for the CMA-ES than in selecting on Un. Next to greater hypervolume per solution, which was already shown on AckleyInfeasible, RE_{mDP} is now thus also able to produce more solutions than Un. The simpler line explore, with its isotropic search distribution, runs into infeasible space after only a small distance was covered in the decision space, deducible from the small number of steps taken (median average of 7) and the used fixed stepsize of 0.01. This leads to result sets containing lots of solutions, but close together, as is visible from Figure 6.4.

Moreover, we determine in what generation the best-performing solution per run is found, see Figure 6.5(c), to rule out that continued exploration is unbeneficial. This generation number turns out to be approximately normally-distributed, with a bias to



(a) Mean Steps per Exp. **(b)** Mean Dist. per Exp. **(c)** Best Sol. in Gen.

Figure 6.5 Analysis of Alternating-restart-scheme Variants. For the 20 runs performed per method, the average number of steps per exploration phase in a run is displayed in (a), and the average distance covered in the decision space per exploration phase in a run is displayed in (b), the latter not being available for line explore. Furthermore, the spread of the generation in which a run's best-quality solution was found is shown in (c). Un and RE_{mDP} cover a similar distance per exploration phase, but Un needs much more generations to do so. Furthermore, the generation in which the best solution is found is roughly normally-distributed.

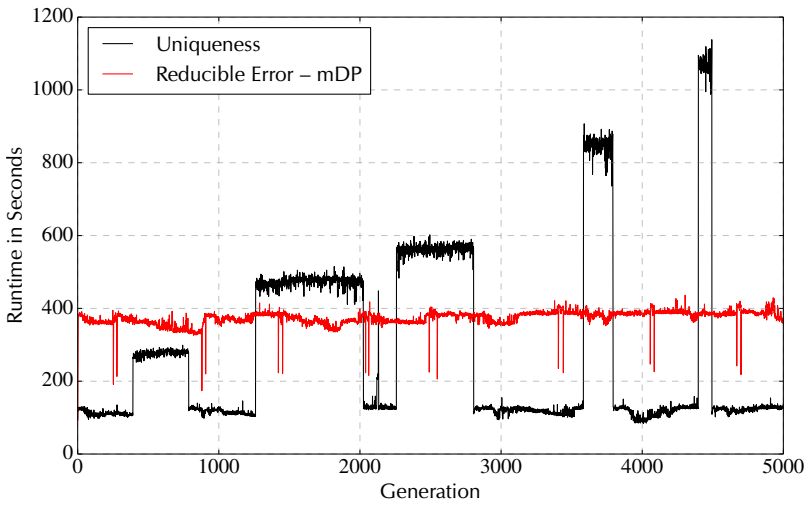
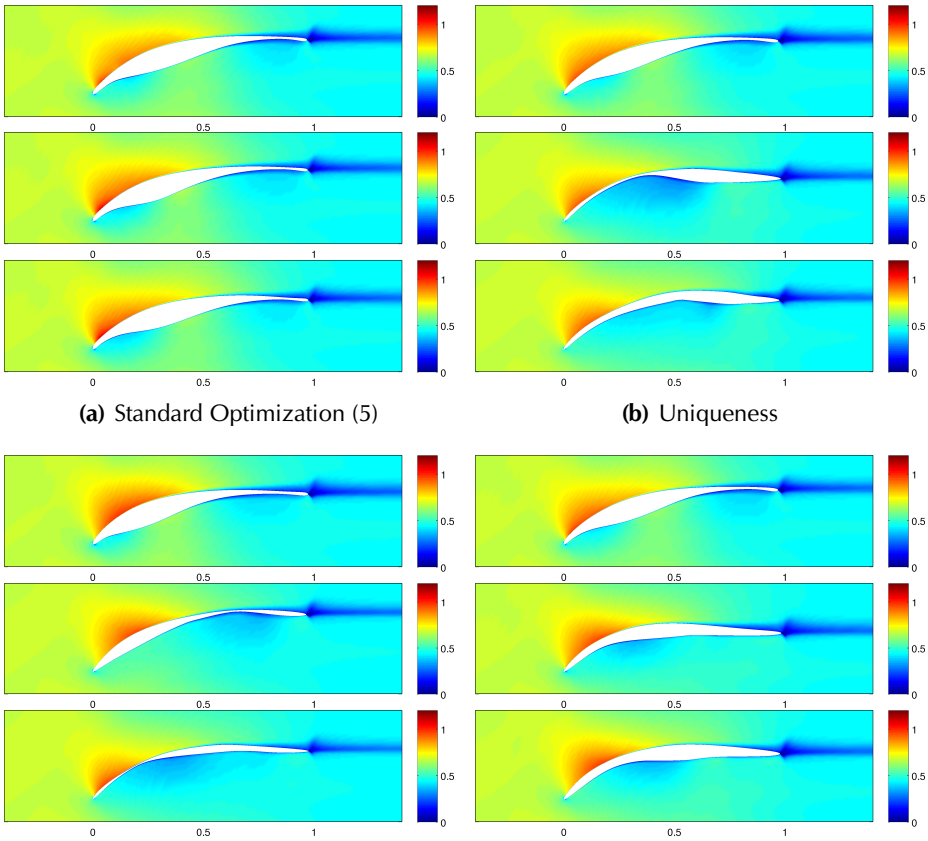


Figure 6.6 Median-run Runtime Analysis. Of an Un and RE_{mDP} run with median hypervolume score, the development of the running time per generation is plotted. Un shows faster quality-optimization phases and increasingly-slower exploration phases as the run continuous. RE_{mDP} shows an equivalent running time for the quality-optimization and exploration phases, depending on the time needed to train the surrogate models used in calculating DP. In the RE_{mDP} run, the start of a new phase is marked by a lower running time, because within a certain phase, only the solutions generated in that phase are used for training. Starting with a quality-optimization phase, it can be seen that the exploration phases in RE_{mDP} are much shorter than in Un.



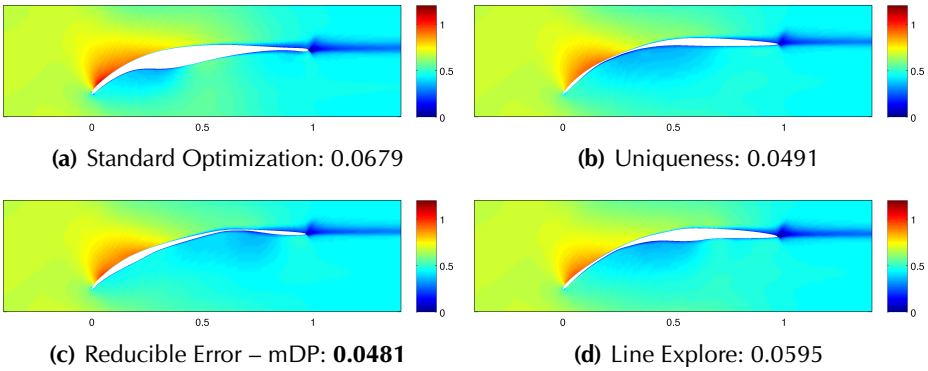
(a) Standard Optimization (5) **(b)** Uniqueness

(c) Reducible Error – mDP **(d)** Line Explore

Figure 6.7 Median-run Found Geometries. Per method, three geometries found in a run with median hypervolume score are displayed. The variants of the alternating restart scheme first converge to a local optimum close to the geometries found by standard optimization, but then generate more-deviating geometries. Nevertheless, all Mach number distributions show high resemblance, seemingly as the left part of the suction side of all blades is similar. Quality values, from top to bottom and to be minimized, of solutions in (a), 0.0838, 0.0796, 0.0784, in (b), 0.0881, 0.0637, 0.0614, in (c), 0.0956, 0.0625, 0.0484, and in (d), 0.0803, 0.0761, 0.0681.

later generations for RE_{mDP} and line explore.

The time to complete one run per method is approximately, using 24 processor cores in parallel per run, 30 hours for a single standard-optimization run, two weeks for Un, three weeks for RE_{mDP}, and one week for line explore. Of a run with median hypervolume score for Un and RE_{mDP}, the runtime is further analyzed in Figure 6.6. For Un, the running time of generations in the quality-optimization phase



(a) Standard Optimization: 0.0679 **(b)** Uniqueness: 0.0491

(c) Reducible Error – mDP: **0.0481** **(d)** Line Explore: 0.0595

Figure 6.8 Best Solution Found. Per method, the best geometry found is displayed. All best solutions found adhere to the thickness constraints: The alternating-restart-scheme variants therefore seem to require adjustment of these constraints, as their best geometries found appear too thin to be feasible in practice.

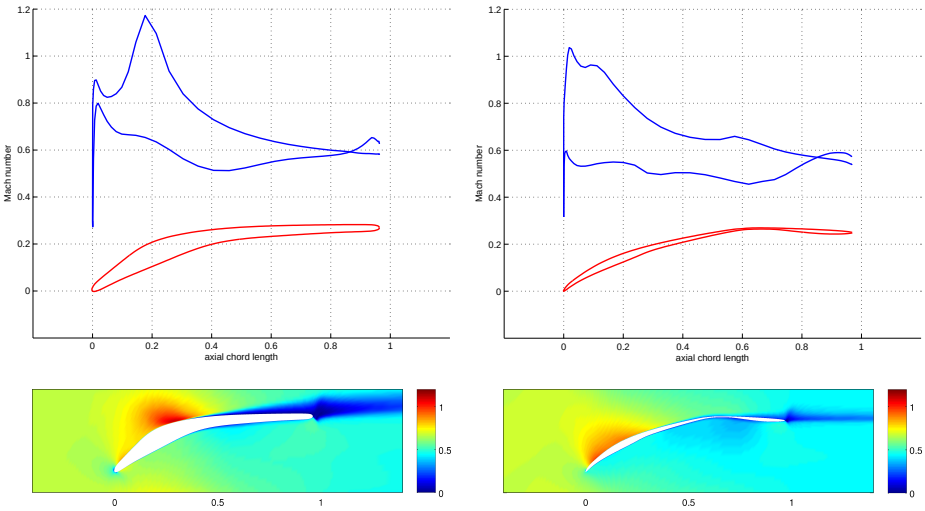
is constant, while for generations in the exploration phase, it strongly increases during the search. This is because determining Un, which entails comparing to all solutions sampled in the quality-optimization phases, becomes more intensive as more solutions have been generated. For RE_{mDP}, most time is used for training the 10 new local models per generation. As the number of training points used is constant while the search continues, so is the running time required for training. The regional error calculation, see Section 4.3.2, per solution in the exploration phase should become more demanding, as this is done taking all previously sampled solutions into account, but this trend is not visible from Figure 6.6. Although Un starts off with lower running time per generation, for runs with more than 5000 generations, RE_{mDP} is recommended based on its nearly constant running time per generation.

For comparing the actual geometries found by the different methods, of the runs with median hypervolume score, three solutions found are displayed in Figure 6.7 (on the previous page). After converging to the default optimum, the alternating-restart-scheme variants find more diverse and better-quality solutions outside of the default basin of attraction. Generally, the thinner the geometries get, the better their quality scores become. Notwithstanding, the distributions of the relative air speed around the airfoils are highly similar, as the variance in the shape of the left part of the suction sides (top side of the blades) is low.

In Figure 6.8, the best geometry found per method is displayed. Given the fact that all these geometries adhere to the minimal-thickness constraints, the way in

Table 6.1 Best Solution Found. Per method, the best solution found is listed. Used notation: Quality value f, pressure loss coefficient ω, outflow angle α_2, leading edge radius r_{le}, trailing edge radius r_{te}, maximum thickness θ_{max}, minimum thickness θ_{min}, and chord length cl. The best values are in bold.

	f	ω $\times 10^{-2}$	α_2	r_{le} % × cl	r_{te} % × cl	θ_{max} % × cl	θ_{min} % × cl
Standard Optimization	0.0679	2.26	0.299°	0.503	0.513	**5.00**	0.503
Uniqueness	0.0491	1.82	0.303°	0.501	0.501	5.01	0.501
Reducible Error – mDP	**0.0481**	**1.79**	**0.277°**	**0.500**	**0.500**	5.02	**0.500**
Line Explore	0.0595	2.07	**0.277°**	0.505	0.504	5.02	0.504
Initial Geometry	132	6.28	−5.96°	0.627	0.643	4.55	0.627



(a) Initial Solution **(b)** Best Solution Found

Figure 6.9 Comparison of Initial and Best Solution Found. The top-row plots show the Mach-number-distribution profile directly on the surface of the airfoils, and the bottom-row plots the Mach number distribution around the airfoils. First of all, the highest Mach-number-peak is lower for the best solution found, and located closer to the leading edge of the airfoil. Moreover, the best solution minimizes all minimal-thickness constraints to their lowest-allowed levels, see Table 6.1. Furthermore, while the outflow angle of the initial solution shows a deviation of −5.96°, in the best solution this is minimized to 0.277°, smaller than the tolerance of 0.3° and thus not degrading the quality score. Related to this outflow angle is the maximum blade thickness, which is determined orthogonally to it. For the best solution's slightly down-pointing outflow, this, clearly undesirably, leads to a *greater* blade thickness of 5.02% × cl than the initial blade's 4.55% × cl.

which the minimum and maximum thickness is determined requires adjustment, as the alternating-restart-scheme found geometries appear too thin to be realized in practice. This is further demonstrated in Table 6.1 and Figure 6.9, when comparing the best solution found to the initial solution from which the search is always started. The best solution, clearly with a smaller maximum thickness, gets a higher maximum-thickness score than the initial solution because of the way in which these are determined, orthogonally to the outflow angle.

6.5 · Summary

This chapter reports on the application of the most-promising exploration measures identified in Chapter 4, as integrated into quality-based search using an *alternating restart scheme* proposed in Chapter 5, to a real-world airfoil-optimization task. It concerns the optimization of a two-dimensional representation of a stator blade involving 16 control points, and hence 32 decision variables. Instances of the stator blade in question are used directly after the low-pressure compressor in a small turbofan engine, for straightening the flow of the compressed air.

After defining a *domain-specific distance measure* to be calculated over the two-dimensional blade geometries, three variants of exploration within the alternating restart scheme were tested on the real-world test case: Distance-based novelty measure *uniqueness* (Un), which utilizes the domain-specific distance measure, interestingness measure *reducible error* ($\mathrm{RE}_{\mathrm{mDP}}$), based on memories of *dispersion-in-prediction* novelty, and the baseline method of exploring in a straight line, for which the gradient is randomly chosen. The instantiations of the alternating restart scheme were compared with the combined outcome of multiple standard optimization runs.

For determining diversity in the result sets of the different methods, the domain-specific distance measure is again used. Utilizing the performance measuring scheme introduced in Chapter 3, appreciating both quality and diversity in a set of solutions, $\mathrm{RE}_{\mathrm{mDP}}$-based exploration was clearly shown to deliver the best-performing method. It outperforms the combined outcome of multiple standard-optimization runs, using the same number of quality evaluations, in the sense that it finds more solutions, more diverse solutions, *and* finds solutions of better quality! Importantly, it manages to do so *in spite of* consuming quality evaluations in its exploration phases, which reduces its budget of evaluations available for the actual quality optimization.

Exploration based on Un provides more diverse and higher-quality solutions than

standard optimization as well, but induces slower-moving exploration than RE_{mDP} in the sense of distance covered in the decision space per generation, thereby resulting in less solutions found. The baseline method of line exploring finds lots of solutions but with poor diversity, because its rudimentary exploration approach only allows for small distances to be traveled before running into infeasible space.

7

Conclusions and Outlook

The main goal of this work is to develop a method that, operating on top of an *Evolutionary Algorithm*, increases the likeliness of finding *innovative solutions*. This likeliness is laid out to be increased with the *diversity* of the solutions found, provided that they are of sufficient quality. The developed method needs to be applicable in a scenario in which the search is required to be started from a single, fixed solution. Therefore, a scheme is envisioned in which the search is performed in a *sequential* fashion, zooming in on a locally-optimal solution, and then exploring for a new potentially high-quality region based on a *memory* of solutions encountered earlier in the search. Two *exploration criteria*, one using an *archive* of earlier solutions as memory and the other deriving from a *surrogate model* trained on earlier solutions, were established to be worthwhile for integration into quality-based search. The resulting schemes were applied to a real-world airfoil optimization task, showing both to perform better than the baseline method of multiple standard optimization runs. The model-based approach delivers the best results, in the sense that it finds more solutions, more diverse solutions, and better-quality solutions than the baseline method.

Next, in Section 7.1, the thesis is summarized in more detail with the most-important conclusions listed, while Section 7.2 provides future research directions.

7.1 · Summary

An innovative solution is a *product design* that has the potential of leading to an *innovation*, that is, the resulting innovative product getting adopted by the end users. *Innovativeness*, the property of being innovative, is defined in this work as being *novel* and of *high-quality*, both with respect to the comprehensive *reference set* of *state-of-the-art* solutions in the application domain. The state-of-the-art solutions are the

established highest-quality solutions, generally regarded as such by the engineering community. Novelty is the distance to the closest solution from the used reference set, with respect to a *domain-specific distance measure*. This domain-specific distance measure is used to isolate and compare on those solution aspects that are relevant to the application.

Automated determination of a solution's true innovative value is complicated because of the need of compiling the comprehensive set of state-of-the-art solutions. On the one hand, it is difficult to take all established high-quality solutions into account, on the other hand, it is not easy to formulate patented solutions in such a way that they can be interpreted and compared to automatically. Instead of searching for innovative solutions directly, we therefore aim to deliver *diverse*, high-quality solutions, where the diversity of a set of solutions is evaluated with respect to the same domain-specific distance measure as is used in determining innovativeness. The assumption is that with increased diversity and quality in a result set, the chance for actual innovative solutions to be present in it increases as well.

The envisioned approach for finding diverse high-quality solutions is through incorporating an exploration criterion into quality-based search, and making the search alternate between *exploitation* (i.e., optimization on quality) and exploration in a sequential pattern. To steer clear of areas that were visited already, exploration needs a memory of where the search was before. As memory, an archive of earlier encountered solutions can be used, or a surrogate model that is trained on the earlier encountered solutions. The resulting *online* novelty of a solution, for usage as exploration criterion, is then either the *distance* to the solutions in the archive (*distance-based novelty*) or the *error* that the surrogate model makes in its *prediction* for it (*learning-based novelty*). The surrogate model approximates the mapping from solution to quality value.

In applying distance-based novelty, it is straightforward that diversity with respect to the same domain-specific distance measure is promoted. In selecting on learning-based novelty, on the other hand, the assumption is that solutions with maximum error improve the surrogate model most upon including them in model training: Learning-based novelty is a predictor for the *learning progress* that a solution leads to. Potentially, by following a pattern of exploration that maximizes learning progress, solutions are generated that are distant from each other with respect to the domain-specific distance measure. Among others depending on the used domain-specific distance measure, possibly, exploration moves through the search space more efficiently in

maximizing learning progress than in selecting on distance-based novelty.

Testing on an artificial function reveals that the simplified distance-based novelty expression that was put forward (*uniqueness*) does not perform worse than the original formulation (*sparseness*): Uniqueness (Un) is the distance to the closest solution in an archive of all generated solutions. Learning-based novelty, on the other hand, turns out to be insufficient as predictor of learning progress: It needs to be extended by an approach that accounts for areas in which the model does not improve, to prevent exploration from stagnating, that is, getting stuck in such an area. The composite approach is termed the *interestingness* of a solution, deriving from learning-based novelty but providing a better prediction of the learning progress based on the *earlier-observed* modeling errors in the *region* in which a solution lies.

Of four tested, *reducible error* (RE) is the best-performing interestingness expression: It substracts an average of earlier-observed errors from the average of recent errors in a certain region, and thereby penalizes regions in which the error stays high. Herein, the assumption is preserved that high (recent) errors lead to high learning progress. For determining the modeling errors, *dispersion in predictions* (DP) is used, a learning-based novelty expression that *estimates* the modeling error by comparing the predictions made by multiple surrogate models. The resulting measure is denoted as $\mathrm{RE}_{\mathrm{mDP}}$, indicating that the observed errors in a region are averaged as a *long-term* and a *short-term memory*, obtained through exponential smoothing at different rates.

Novelty and interestingness express deviation from available knowledge, not the chance for high quality. This is in line with the view of exploration and exploitation having inherently conflicting dynamics: While exploring, the search *diverges*, and in exploiting, it *converges*. In including an exploration criterion in quality-based search, we therefore strictly separate exploration from exploitation. Quality-based optimization is run until convergence, i.e., the search distribution from which new points are generated has reached an indicated minimum magnitude, after which we explore for a new starting point for quality-based optimization. Through their divergent nature, it is not straightforward to formulate a similar *stopping condition* for the exploration phases.

However, the real-world application on which the developed methods are intended to be applied includes large regions with *intolerable* solutions. Of these "*infeasible* regions", the boundaries are unknown in advance. Therefore, the search is required to be started from a basic solution that is known to be feasible. On the other hand,

we *can* conveniently use entering infeasible space as a stopping condition for an exploration phase, in which the assumption is that this is sufficient for leaving the *basin of attraction* of the *optimum* that quality optimization converged to prior to start of the exploration phase.

The intended real-world application involves the optimization of a *stator blade* design in a small *turbofan* engine that is intended for propelling small business jet aircraft. In a *gas turbine*, non-moving stator blades are installed after the moving *rotor blades* to straighten the air flow. The quality of a solution, which is represented as a vector containing 32 variables, is approximated using *aerodynamic simulation*. As such, establishing this quality value is computationally intensive. After defining a domain-specific distance measure, deriving from the two-dimensional stator-blade *profiles* that are obtained by decoding the vector representation, we apply distance-based novelty expression Un and interestingness expression $\mathrm{RE}_{\mathrm{mDP}}$ using the alternating quality-optimization/exploration scheme described above. Next to usage in Un, the domain-specific distance measure is required for determining the diversity of the produced result sets, to compare performance between methods.

As baseline method, multiple standard quality-based optimization runs are used. Five of such runs account for the same quality-evaluation budget as is used for an exploration-assisted run, and can hence be seen as a single run that has been *restarted* four times. Assisted by interestingness-based exploration, the *CMA-ES*, an Evolutionary Algorithm with advanced online adaptation of its search distribution, delivers more solutions, more diverse solutions, *and* solutions of greater quality than five of the unassisted CMA-ES runs combined. Exploration based on Un provides more diverse and higher-quality solutions than standard optimization as well, but induces slower-moving exploration than $\mathrm{RE}_{\mathrm{mDP}}$ and thereby results in less solutions found, which influences the diversity scores and, implicitly, the best quality value found. As was surmised, in exploration based on learning progress, it can occur that the underlying optimizer, i.e., here the CMA-ES, is presented with a search landscape that it can more-efficiently traverse than the landscape derived from the domain-specific distance between solutions.

7.2 · Outlook

The interestingness-based exploration variant, though providing the best results, is an involved method, relying on multiple layers of components. Implementing it is complicated, with the risk of easily making mistakes in the process. It should be examined which parts are essential in its performance, and whether a similar induced pattern of exploration can be attained using a less complex setup. This examination was started using the *line-explore* method, which simply explores in a randomly-chosen straight line through the search space, but which was clearly outperformed by both Un and RE_{mDP}. However, in this setup, the gradient for the line was chosen from an *isotropic* probability distribution, parameterized-equally in all dimensions, which can possibly be done more cleverly by taking the adapted search distribution of the CMA-ES into account. When considering to apply the methods presented in this thesis, it is recommend to start with the less-performing, but simpler to implement archive-based exploration variant Un. Nevertheless, despite being more complex, on the real-world application, RE_{mDP} *is* computationally less expensive than the Un scheme, as the latter requires comparing to all earlier generated solutions using the domain-specific distance measure.

Furthermore, the integration scheme by which both exploration variants are introduced into quality-based search depends on regions of intolerable solutions to exist in the search space, serving as stopping condition for the exploration when such an area is entered. To make the developed measures applicable to problems without similar infeasible regions, alternative stopping conditions are to be examined, for instance based on the development of the quality values encountered while exploring.

Lastly, in the real-world airfoil application, domain-specific distance is ideally derived from the *flow-field* approximations that are obtained through simulation and from which the quality values are determined. Ways of using these approximations, which are available as two-dimensional color plots, to compare solutions are to be studied, for instance by *morphing* the blade profile in one such image into a profile in another image, and then determining the difference between the resulting images pixel-wisely.

Bibliography

[**Ackley, 1987**] Ackley, D. H. (1987). *A Connectionist Machine for Genetic Hillclimbing*. Springer.
Cited on page 77.

[**Arima et al., 1999**] Arima, T., Sonoda, T., Shirotori, M., Tamura, A., and Kikuchi, K. (1999). A Numerical Investigation of Transonic Axial Compressor Rotor Flow Using a Low-Reynolds-Number $k-\varepsilon$ Turbulence Model. *Turbomachinery*, 121(1):44–58.
Cited on page 101.

[**Auger and Hansen, 2005**] Auger, A. and Hansen, N. (2005). A Restart CMA Evolution Strategy with Increasing Population Size. In *2005 IEEE Congress on Evolutionary Computation, CEC 2005*, pages 1769–1776. IEEE Computer Society.
Cited on pages 79 and 85.

[**Bäck, 1996**] Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press.
Cited on pages 6 and 77.

[**Baranes and Oudeyer, 2013**] Baranes, A. and Oudeyer, P.-Y. (2013). Active Learning of Inverse Models with Intrinsically Motivated Goal Exploration in Robots. *Robotics and Autonomous Systems*, 61(1):49–73.
Cited on pages 37, 58, 59, and 65.

[**Barto and Dietterich, 2004**] Barto, A. and Dietterich, T. (2004). Reinforcement Learning and Its Relationship to Supervised Learning. In *Handbook of Learning and Approximate Dynamic Programming*, pages 45–63. Wiley-IEEE Press.
Cited on page 37.

[**Cox and John, 1997**] Cox, D. and John, S. (1997). SDO: A Statistical Method for Global Optimization. In *Multidisciplinary Design Optimization: State of the Art*, pages 315–329. Siam.
Cited on page 16.

[**Cuccu and Gomez, 2011**] Cuccu, G. and Gomez, F. (2011). When Novelty is Not Enough. In *Applications of Evolutionary Computation: Evo* 2011*, volume 6624 of *LNCS*, pages 234–243. Springer.
Cited on pages 42, 80, and 81.

[**Cuccu et al., 2011a**] Cuccu, G., Gomez, F., and Glasmachers, T. (2011a). Novelty-based Restarts for Evolution Strategies. In *2011 IEEE Congress on Evolutionary Computation, CEC 2011*, pages 158–163. IEEE Computer Society.
Cited on pages 37, 80, 85, and 86.

[**Cuccu et al., 2011b**] Cuccu, G., Luciw, M., Schmidhuber, J., and Gomez, F. (2011b). Intrinsically Motivated NeuroEvolution for Vision-based Reinforcement Learning. In *2011 IEEE International Conference on Development and Learning, ICDL 2011*. IEEE Press.
Cited on pages 37 and 46.

[**Deb, 2013**] Deb, K. (2013). *Innovization: Discovering Innovative Solution Principles through Optimization*. Springer.
Cited on page 19.

[**Deb et al., 2002**] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.
Cited on page 82.

[**Deb and Srinivasan, 2006**] Deb, K. and Srinivasan, A. (2006). Innovization: Innovating Design Principles through Optimization. In *8th Annual Conference on Genetic and Evolutionary*

Computation, GECCO'06, pages 1629–1636. ACM Press.
Cited on page 19.

[**Eiben and Smith, 2003**] Eiben, A. and Smith, J. (2003). *Introduction to Evolutionary Computing*. Springer.
Cited on page 6.

[**Emmerich, 2005**] Emmerich, M. (2005). *Single- and Multi-objective Evolutionary Design Optimization Assisted by Gaussian Random Field Metamodels*. PhD thesis, Department of Computer Science, TU Dortmund, Germany.
Cited on pages 12, 16, and 17.

[**Emmerich et al., 2013**] Emmerich, M., Deutz, A., and Kruisselbrink, J. (2013). On Quality Indicators for Black-box Level Set Approximation. In *EVOLVE — A Bridge between Probability, Set-oriented Numerics, and Evolutionary Computation*, volume 447 of *SCI*, pages 157–185. Springer.
Cited on page 89.

[**Emmerich et al., 2012**] Emmerich, M., Deutz, A., Kruisselbrink, J., and Reehuis, E. (2012). Evolutionary Level Set Approximation. In *EVOLVE 2012, A Bridge between Probability, Set-oriented Numerics, and Evolutionary Computation*.
Cited on page 89.

[**Emmerich et al., 2006**] Emmerich, M., Giannakoglou, K., and Naujoks, B. (2006). Single- and Multi-objective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439.
Cited on pages 15, 16, 50, 81, and 83.

[**Forrester et al., 2008**] Forrester, A., Sóbester, A., and Keane, A. (2008). *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley.
Cited on page 14.

[**Freeman, 1991**] Freeman, C. (1991). The Nature of Innovation and the Evolution of the Productive System. In *Technology and Productivity, The Challenge for Economic Policy*, volume 22, pages 303–314. OECD.
Cited on page 26.

[**Garcia and Calantone, 2002**] Garcia, R. and Calantone, R. (2002). A Critical Look at Technological Innovation Typology and Innovativeness Terminology: A Literature review. *Product Innovation Management*, 19(2):110–132.
Cited on pages 19, 21, 22, and 26.

[**Geng and Hamilton, 2006**] Geng, L. and Hamilton, H. (2006). Interestingness Measures for Data Mining: A Survey. *ACM Computing Surveys*, 38(3).
Cited on page 38.

[**Gero and Saunders, 2000**] Gero, J. and Saunders, R. (2000). Constructed Representations and Their Functions in Computational Models of Designing. In *Fifth Conference on Computer-aided Architectural Design Research in Asia, CAADRIA 2000*, pages 215–224.
Cited on page 37.

[**Graening et al., 2010**] Graening, L., Aulig, N., and Olhofer, M. (2010). Towards Directed Open-ended Search by a Novelty Guided Evolution Strategy. In *11th International Conference on Parallel Problem Solving from Nature, PPSN XI*, volume 6239 of *LNCS*, pages 71–80. Springer.
Cited on pages 37, 46, 48, 77, 80, and 83.

[**Graziano et al., 2011**] Graziano, V., Glasmachers, T., Schaul, T., Pape, L., Cuccu, G., Leitner, J., and Schmidhuber, J. (2011). Artificial Curiosity for Autonomous Space Exploration. *Acta Futura*, 4:41–51.
Cited on pages 20, 21, and 37.

[**Hansen, 2011**] Hansen, N. (2011). The CMA Evolution Strategy: A Tutorial. Technical report, Laboratoire de Recherche en Informatique (LRI), Université Paris-Sud, France.
Cited on pages 7 and 10.

[**Hansen and Kern, 2004**] Hansen, N. and Kern, S. (2004). Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In *8th International Conference on Parallel Problem Solving from Nature, PPSN VIII*, volume 3242 of *LNCS*, pages 282–291. Springer.
Cited on pages 7, 9, and 10.

[**Hansen and Ostermeier, 2001**] Hansen, N. and Ostermeier, A. (2001). Completely Derandomized Self-adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195.
Cited on pages 7 and 10.

[**Hasenjäger et al., 2005**] Hasenjäger, M., Sendhoff, B., Sonoda, T., and Arima, T. (2005). Three Dimensional Evolutionary Aerodynamic Design Optimization with CMA-ES. In *7th Annual Conference on Genetic and Evolutionary Computation, GECCO'05*, pages 2173–2180. ACM Press.
Cited on page 102.

[**Haykin, 1998**] Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition.
Cited on pages 11 and 12.

[**Hébert and Crémilleux, 2007**] Hébert, C. and Crémilleux, B. (2007). A Unified View of Objective Interestingness Measures. In *MLDM 2007*, volume 4571 of *LNAI*, pages 533–547. Springer.
Cited on page 38.

[**Hester, 2012**] Hester, T. (2012). *TEXPLORE: Temporal Difference Reinforcement Learning for Robots and Time-constrained Domains*. PhD thesis, The University of Texas at Austin, USA.
Cited on pages 45 and 52.

[**Hester and Stone, 2012**] Hester, T. and Stone, P. (2012). Intrinsically Motivated Model Learning for a Developing Curious Agent. In *2012 IEEE International Conference on Development and Learning, ICDL 2012*. IEEE Press.
Cited on pages 37, 42, 45, 52, 53, and 75.

[**Hilderman and Hamilton, 2003**] Hilderman, R. and Hamilton, H. (2003). Measuring the Interestingness of Discovered Knowledge: A Principled Approach. *Intelligent Data Analysis*, 7(4):347–382.
Cited on page 38.

[**Igel and Hüsken, 2003**] Igel, C. and Hüsken, M. (2003). Empirical Evaluation of the Improved Rprop Learning Algorithms. *Neurocomputing*, 50:105–123.
Cited on pages 12, 48, and 105.

[**Jin et al., 2002**] Jin, Y., Olhofer, M., and Sendhoff, B. (2002). A Framework for Evolutionary Optimization with Approximate Fitness Functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494.
Cited on page 15.

[**Jin and Sendhoff, 2004**] Jin, Y. and Sendhoff, B. (2004). Reducing Fitness Evaluations Using Clustering Techniques and Neural Network Ensembles. In *6th Annual Conference on Genetic and Evolutionary Computation, GECCO'04*, volume 3102 of *LNCS*, pages 688–699. Springer.
Cited on pages 48, 52, and 53.

[**Jones et al., 1998**] Jones, D., Schonlau, M., and Welch, W. (1998). Efficient Global Optimization of Expensive Black-box Functions. *Global Optimization*, 13(4):455–492.
Cited on pages 12, 15, 17, 50, and 81.

[**Kleijnen and Mehdad, 2012**] Kleijnen, J. and Mehdad, E. (2012). Kriging in Multi-response Simulation, Including a Monte Carlo Laboratory. Technical Report 2012-039, CentER, TiSEM, Tilburg University, The Netherlands.
Cited on page 14.

[**Kruisselbrink, 2012**] Kruisselbrink, J. (2012). *Evolution Strategies for Robust Optimization*. PhD thesis, Leiden Institute of Advanced Computer Science (LIACS), Leiden University, The Netherlands.
Cited on pages 6, 7, 38, 55, and 101.

[**Kushner, 1962**] Kushner, H. (1962). A Versatile Stochastic Model of a Function of Unknown and Time Varying Form. *Mathematical Analysis and Applications*, 5(1):150–167.
Cited on page 15.

[**Lehman and Stanley, 2008**] Lehman, J. and Stanley, K. (2008). Exploiting Open-endedness to Solve Problems through the Search for Novelty. In *Artificial Life XI: Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 329–336. MIT Press.
Cited on pages 42 and 43.

[**Lehman and Stanley, 2011**] Lehman, J. and Stanley, K. (2011). Abandoning Objectives: Evolution through the Search for Novelty Alone. *Evolutionary Computation*, 19(2):189–223.
Cited on pages 37 and 43.

[**Lehman et al., 2013**] Lehman, J., Stanley, K., and Miikkulainen, R. (2013). Effective Diversity Maintenance in Deceptive Domains. In *15th Annual Conference on Genetic and Evolutionary Computation, GECCO'13*, pages 215–222. ACM Press.
Cited on pages 31, 42, and 83.

[**Lim et al., 2010**] Lim, D., Jin, Y., Ong, Y.-S., and Sendhoff, B. (2010). Generalizing Surrogate-assisted Evolutionary Computation. *IEEE Transactions on Evolutionary Computation*, 14(3):329–355.
Cited on page 48.

[**Loshchilov et al., 2012**] Loshchilov, I., Schoenauer, M., and Sebag, M. (2012). Self-adaptive Surrogate-assisted Covariance Matrix Adaptation Evolution Strategy. In *14th Annual Conference on Genetic and Evolutionary Computation, GECCO'12*, pages 321–328. ACM Press.
Cited on pages 6 and 47.

[**Lu et al., 2013**] Lu, J., Li, B., and Jin, Y. (2013). An Evolution Strategy Assisted by an Ensemble of Local Gaussian Process Models. In *15th Annual Conference on Genetic and Evolutionary Computation, GECCO'13*, pages 447–454. ACM Press.
Cited on page 81.

[**Luciw et al., 2011**] Luciw, M., Graziano, V., Ring, M., and Schmidhuber, J. (2011). Artificial Curiosity with Planning for Autonomous Perceptual and Cognitive Development. In *2011 IEEE International Conference on Development and Learning, ICDL 2011*. IEEE Press.
Cited on pages 37, 58, 59, 60, and 67.

[**Mockus et al., 1978**] Mockus, J., Tiesis, V., and Zilinskas, A. (1978). The Application of Bayesian Methods for Seeking the Extremum. In *Towards Global Optimization 2*, pages 117–129. North-Holland.
Cited on page 17.

[**Mouret, 2011**] Mouret, J.-B. (2011). Novelty-based Multiobjectivization. In *2009 EvoDeRob Workshop*, volume 341 of *SCI*, pages 139–154. Springer.
Cited on pages 42, 80, and 83.

[**Olhofer et al., 2002**] Olhofer, M., Arima, T., Sonoda, T., Fischer, M., and Sendhoff, B. (2002). Aerodynamic Shape Optimisation Using Evolutionary Strategies. In *Optimization in Industry*, pages 83–94. Springer.
Cited on page 99.

[**Oudeyer and Kaplan, 2007**] Oudeyer, P.-Y. and Kaplan, F. (2007). What is Intrinsic Motivation? A Typology of Computational Approaches. *Frontiers in Neurorobotics*, 1(6):1–14.
Cited on pages 37, 46, 48, 60, 61, and 62.

[**Oudeyer et al., 2007**] Oudeyer, P.-Y., Kaplan, F., and Hafner, V. (2007). Intrinsic Motivation Systems for Autonomous Mental Development. *IEEE Transactions on Evolutionary Computation*,

11(2):265–286.
Cited on pages 37, 49, 58, 59, 61, 62, and 63.

[**Padmanabhan and Tuzhilin, 1998**] Padmanabhan, B. and Tuzhilin, A. (1998). A Belief-driven Method for Discovering Unexpected Patterns. In *4th International Conference on Knowledge Discovery and Data Mining, KDD 1998*. American Association for Artificial Intelligence.
Cited on page 38.

[**Rasmussen and Williams, 2006**] Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press.
Cited on pages 13 and 50.

[**Reehuis et al., 2011**] Reehuis, E., Kruisselbrink, J., Olhofer, M., Graening, L., Sendhoff, B., and Bäck, T. (2011). Model-guided Evolution Strategies for Dynamically Balancing Exploration and Exploitation. In *10th International Conference on Artificial Evolution, EA 2011*, pages 306–317.
Cited on pages 3, 4, 47, 52, 53, 77, and 83.

[**Reehuis et al., 2013a**] Reehuis, E., Olhofer, M., Emmerich, M., Sendhoff, B., and Bäck, T. (2013a). Novelty and Interestingness Measures for Design-space Exploration. In *15th Annual Conference on Genetic and Evolutionary Computation, GECCO'13*, pages 1541–1548. ACM Press.
Cited on pages 3, 53, and 58.

[**Reehuis et al., 2013b**] Reehuis, E., Olhofer, M., Sendhoff, B., and Bäck, T. (2013b). Novelty-guided Restarts for Diverse Solutions in Optimization of Airfoils. In *EVOLVE 2013, A Bridge between Probability, Set-oriented Numerics, and Evolutionary Computation*, pages 25–30. LIACS, Leiden University.
Cited on pages 3, 4, and 86.

[**Reehuis et al., 2013c**] Reehuis, E., Olhofer, M., Sendhoff, B., and Bäck, T. (2013c). Learning-guided Exploration in Airfoil Optimization. In *14th International Conference on Intelligent Data Engineering and Automated Learning, IDEAL'13*, volume 8206 of *LNCS*, pages 506–513. Springer.
Cited on pages 3, 4, 53, 58, and 86.

[**Riedmiller and Braun, 1993**] Riedmiller, M. and Braun, H. (1993). A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In *1993 IEEE International Conference on Neural Networks, ICNN'93*, pages 586–591. IEEE Press.
Cited on page 12.

[**Rogers, 2010**] Rogers, E. (2010). *Diffusion of Innovations*. Simon and Schuster.
Cited on pages 19 and 26.

[**Rückstieß et al., 2010**] Rückstieß, T., Sehnke, F., Schaul, T., Wierstra, D., Sun, Y., and Schmidhuber, J. (2010). Exploring Parameter Space in Reinforcement Learning. *Paladyn*, 1(1):14–24.
Cited on page 37.

[**Rumelhart et al., 1986**] Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536.
Cited on page 12.

[**Saunders, 2002**] Saunders, R. (2002). *Curious Design Agents and Artificial Creativity: A Synthetic Approach to the Study of Creative Behaviour*. PhD thesis, Department of Architectural and Design Science, Faculty of Architecture, University of Sydney, Australia.
Cited on pages 20 and 38.

[**Saunders and Gero, 2001**] Saunders, R. and Gero, J. (2001). Designing for Interest and Novelty. In *Computer Aided Architectural Design Futures 2001*, pages 725–738. Springer.
Cited on page 24.

[**Saunders and Gero, 2004**] Saunders, R. and Gero, J. (2004). Curious Agents and Situated Design Evaluations. *AI EDAM: Artificial Intelligence for Engineering Design, Analysis and*

Manufacturing, 18(2):153–161.
Cited on page 37.

[**Schaul et al., 2011**] Schaul, T., Sun, Y., Wierstra, D., Gomez, F., and Schmidhuber, J. (2011). Curiosity-driven Optimization. In *2011 IEEE Congress on Evolutionary Computation, CEC 2011*, pages 1343–1349. IEEE Computer Society.
Cited on pages 37, 50, and 83.

[**Schmidhuber, 1991**] Schmidhuber, J. (1991). Curious Model-building Control Systems. In *1991 International Joint Conference on Neural Networks, IJCNN'91*, pages 1458–1463. IEEE Press.
Cited on pages 37 and 61.

[**Schmidhuber, 1997**] Schmidhuber, J. (1997). What's Interesting? Technical Report IDSIA-35-97, IDSIA, Lugano, Switzerland.
Cited on pages 21, 24, and 37.

[**Schmidhuber, 2009**] Schmidhuber, J. (2009). Driven by Compression Progress: A Simple Principle Explains Essential Aspects of Subjective Beauty, Novelty, Surprise, Interestingness, Attention, Curiosity, Creativity, Art, Science, Music, Jokes. In *4th International Workshop on Anticipatory Behavior in Adaptive Learning Systems, ABiALS 2008*, volume 5499 of *LNAI*, pages 48–76. Springer.
Cited on page 37.

[**Settles, 2010**] Settles, B. (2010). Active Learning Literature Survey. Technical Report 1648 (Computer Sciences), University of Wisconsin–Madison, USA.
Cited on page 37.

[**Shir et al., 2010**] Shir, O., Emmerich, M., and Bäck, T. (2010). Adaptive Niche Radii and Niche Shapes Approaches for Niching with the CMA-ES. *Evolutionary Computation*, 18(1):97–126.
Cited on pages 75 and 76.

[**Silberschatz and Tuzhilin, 1996**] Silberschatz, A. and Tuzhilin, A. (1996). What Makes Patterns Interesting in Knowledge Discovery Systems. *IEEE Transactions of Knowledge and Data Engineering*, 8(6):970–974.
Cited on page 38.

[**Sóbester et al., 2004**] Sóbester, A., Leary, S., and Keane, A. (2004). A Parallel Updating Scheme for Approximating and Optimizing High Fidelity Computer Simulations. *Structural and Multidisciplinary Optimization*, 27(5):371–383.
Cited on page 15.

[**Solow and Polasky, 1994**] Solow, A. and Polasky, S. (1994). Measuring Biological Diversity. *Environmental and Ecological Statistics*, 1(2):95–103.
Cited on pages 29 and 30.

[**Stein, 1999**] Stein, M. (1999). *Statistical Interpolation of Spatial Data: Some Theory for Kriging*. Springer.
Cited on page 13.

[**Storn and Price, 1997**] Storn, R. and Price, K. (1997). Differential Evolution — A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Global Optimization*, 11(4):341–359.
Cited on pages 14 and 51.

[**Sutton and Barto, 1998**] Sutton, R. and Barto, A. (1998). *Reinforcement Learning*. MIT Press.
Cited on page 37.

[**Tanton, 2005**] Tanton, J. (2005). *Encyclopedia of Mathematics*. Facts On File.
Cited on page 82.

[**Thrun, 1995**] Thrun, S. (1995). Exploration in Active Learning. In *Handbook of Brain Science and Neural Networks*. MIT Press.
Cited on pages 46 and 49.

[**Ulrich et al., 2010**] Ulrich, T., Bader, J., and Thiele, L. (2010). Defining and Optimizing Indicator-based Diversity Measures in Multiobjective Search. In *11th International Conference on Parallel Problem Solving from Nature, PPSN XI*, volume 6238 of *LNCS*, pages 707–717. Springer.
Cited on page 29.

[**Ulrich and Thiele, 2011**] Ulrich, T. and Thiele, L. (2011). Maximizing Population Diversity in Single-objective Optimization. In *13th Annual Conference on Genetic and Evolutionary Computation, GECCO'11*, pages 641–648. ACM Press.
Cited on pages 29 and 30.

[**Xiong et al., 2007**] Xiong, Y., Chen, W., Apley, D., and Ding, X. (2007). A Non-stationary Covariance-based Kriging Method for Metamodelling in Engineering Design. *Numerical Methods in Engineering*, 71(6):733–756.
Cited on page 50.

[**Zitzler and Thiele, 1998**] Zitzler, E. and Thiele, L. (1998). Multiobjective Optimization Using Evolutionary Algorithms — A Comparative Case Study. In *5th International Conference on Parallel Problem Solving from Nature, PPSN V*, volume 1498 of *LNCS*, pages 292–301. Springer.
Cited on page 31.

Samenvatting

Binnen de *optimalisatie* is het doel een *invoer* te vinden waarvoor een gegeven wiskundige functie de *optimale uitvoer* levert. Deze optimale uitvoer is de maximale dan wel de minimale waarde die de functie kan retourneren, een keuze die vooraf gespecificeerd is. Een *ontwerpprobleem* in de reële wereld, zoals het bepalen van de vorm van een vleugelprofiel, kan gezien worden als een geval van *kwaliteitsoptimalisatie*. De invoer is een voorgesteld ontwerp en de uitvoer diens vastgestelde kwaliteit. Aan een ontwerp refereert men dan als *oplossing* voor het optimalisatieprobleem.

Om een optimalisatieprobleem geautomatiseerd te kunnen behandelen moet de invoer op zo'n wijze aangeleverd, dat wil zeggen *gecodeerd*, worden dat het mogelijk is automatisch de oplossingen te variëren en aan te passen. Een oplossing is bijvoorbeeld gecodeerd als een rij getallen, waarna het daadwerkelijke ontwerp verkregen wordt door deze rij getallen te *decoderen*. Verder moet de kwaliteit van een ontwerp ook getalsmatig uitgedrukt worden. Vaak wordt een opzet gebruikt waarin de numeriek uitgedrukte oplossing ingevoerd wordt in een *simulator* die het reële gedrag van het bedoelde ontwerp benadert. Gebruikmakend van een gesimuleerde omgeving komt deze tot een benadering van de kwaliteit. De simulator neemt hier dus de rol aan van een functie die de kwaliteit uitdrukt.

Gegeven de numerieke voorstelling van oplossingen en de beschikbare geautomatiseerde kwaliteitsbepaling kan een *optimalisatiealgoritme* toegepast worden dat, zonder menselijke tussenkomst, de optimale oplossing zoekt door middel van het systematisch variëren van de numerieke invoer. *Evolutionaire Algoritmen* (EA's) zijn methoden die een functie optimaliseren gebruikmakend van een *populatie* van, doorgaans numeriek voorgestelde, oplossingen die *evolueren* richting de optimale configuratie. In tegenstelling tot klassieke, *deterministische* optimalisatiealgoritmen, berusten de *stochastische* EA's op *toeval*, zoals evolutie in de natuur berust op toeval. De functie die geoptimaliseerd dient te worden verzorgt feedback over "*fitness*", in de zin van aangepastheid, van oplossingen. Deze fitness wordt gebruikt om de betere oplossingen van de rest te scheiden. Gezien het feit dat ze werken door middel van een populatie van oplossingen zijn EA's relatief langzaam in het benaderen van een optimum, met betrekking tot het vereiste aantal functie-evaluaties. Ze zijn echter wel redelijk

opgewassen tegen bepaalde verraderlijke eigenschappen in een fitnesslandschap die andere methoden op het verkeerde spoor kunnen zetten. Dit, en de hoge flexibiliteit die EA's hebben in de exacte formulering van het optimalisatieprobleem maakt dat ze vaak worden gebruikt voor ontwerpproblemen.

Optimalisatiemethoden in het algemeen, en EA's in het bijzonder, zijn in staat ondersteuning te bieden bij het vinden van *innovatieve oplossingen*, dit is ruwweg gedefinieerd als een vernieuwende manier om tot goede prestaties of kwaliteit te komen. Een optimalisatiemethode volgt namelijk mogelijk een ander pad door de zoekruimte dan een ingenieur ooit zou doen gebaseerd op diens genoten opleiding, maar zelfs in een EA convergeert de populatie met hoge waarschijnlijkheid naar één enkele geoptimaliseerde oplossing. Door hun stochastische opzet zijn EA's daarentegen wel in staat om alternatieve oplossingen te produceren als ze meerdere malen uitgevoerd worden.

Dit proefschrift heeft als doel een methode te ontwikkelen die maakt dat optimalisatiealgoritmen zich expliciet richten op het produceren van verschillende hogekwaliteitsoplossingen. Het idee is het optimalisatiealgoritme zoals gebruikelijk te laten convergeren naar een optimum, waarna een verkenningsfase start waarin gezocht wordt naar een nieuw startpunt voor de volgende kwaliteitsoptimalisatiefase. Deze sequentiële opzet is vereist gezien het uiteindelijke ontwerpprobleem waarop de methode zal worden getest. Hierin wordt de zoektocht namelijk altijd gestart vanaf dezelfde initiële oplossing, omdat er grote, onduidelijk begrensde gebieden zijn in de zoekruimte met oplossingen waarvoor de simulator geen bruikbare uitkomst produceert.

Voor een mogelijke *exploratiemaat* om te verkenningsfase te sturen richten we ons op de ontwikkelingsrobotica: hierin worden zoekruimten van mogelijke acties doorzocht met als doel de vaardigheden van de robot te verbeteren door middel van het uitvoeren van nieuwe acties. Centraal hierin is het model van de werkelijkheid dat de robot stapsgewijs opbouwt en dat hem in staat stelt vaardigheden ten toon te spreiden. Dit *leren* door middel van acties wordt gedaan naar analogie van hoe mensen en dieren leren: de grens wordt opgezocht van de beschikbare kennis en het opgebouwde begrip want dat is uitdagend, maar indien iets te ver afwijkt van het bekende wordt het te ingewikkeld en verliest het zijn "*interessantheid*".

Deze interessantheid, uitgedrukt met betrekking tot een *benaderingsmodel* van de kwaliteitsfunctie dat opgebouwd wordt tijdens de zoektocht op basis van getoetste

oplossingen, gebruiken we om de verkenningsfase te sturen. In een denkbeeldig zoeklandschap met pieken en dalen beweegt de zoektocht zich in de optimalisatiefase omlaag, want we spreken af dat we op zoek zijn naar minima, terwijl hij in de verkenningsfase weer omhoog beweegt, mogelijk over meerdere pieken en door meerdere dalen.

Het uiteindelijke probleem betreft het ontwerp van een *statorblad* voor toepassing in een turbofan vliegtuigmotor voor kleine zakenjets. De niet-bewegende statorbladen worden achter bewegende rotorbladen geplaatst om de luchtstroom te ontwarren, dat wil zeggen weer recht te maken. Zoals vermeld is de zoekruimte van dit probleem onderhevig aan grote gebieden met ongeschikte oplossingen. Dit kunnen we echter uitbuiten als *stopconditie* voor de verkenningsfase. Zodra we een gebied met ontoelaatbare oplossingen betreden, stoppen we de verkenning en starten we de optimalisatiefase. De kwaliteitsoptimalisatie is in staat het gebied met ontoelaatbare oplossingen op eigen kracht te verlaten. Als stopconditie voor de optimalisatiefase wordt een bepaalde nauwkeurigheid in de bereikte waarden gebruikt, wat mogelijk is gezien het convergente karakter van de kwaliteitsoptimalisatie. Dit alles leidt tot een samengestelde aanpak waarin meer oplossingen worden gevonden, een hogere diversiteit aan oplossingen wordt bereikt en waarin de gevonden oplossingen van betere kwaliteit zijn dan in de standaard aanpak. Deze standaard aanpak is simpelweg het eerdergenoemde meerdere malen uitvoeren van het optimalisatiealgoritme, zonder enige hulp van een exploratiemaat.

Acknowledgments

Here, I would like to thank Dr. Markus Olhofer, my tutor at HRI-EU, for the good (and lengthy..) discussions, but most of all, the pleasant cooperation over the past years. Danke schön, Markus.

About the Author

Edgar Reehuis was born on the 21st of June, 1982, in Breda. In 2001, he started his studies in Computer Science at the Leiden Institute of Advanced Computer Science (LIACS). For his Bachelor's thesis, he investigated design optimization of water distribution networks, the drinking-water backbones of urban regions, using multiobjective Evolutionary Algorithms, under supervision of Dr. Michael Emmerich.

Starting his Master's in 2008, in the track Computer Science Theory and Advanced Technologies of the MSc programme at LIACS, he pursued a research project on automated optimization of warehouse designs under supervision of Prof. dr. Thomas Bäck, a cooperation with the Embedded Systems Institute in Eindhoven and Vanderlande Industries. For his Master's thesis, he returned to a familiar subject, now investigating robustness to sources of environmental uncertainty in multiobjective optimization of water distribution networks, and the added value of explicit methods of accounting for robustness in this application.

After obtaining his Master's degree in 2010 (*cum laude*), Prof. Bäck offered him a PhD position under joined supervision with Prof. dr. Bernhard Sendhoff at the Honda Research Institute Europe (HRI-EU) in Offenbach, near Frankfurt, which offered a nice opportunity to continue working in this field of science, and to gain experience in cooperating with industry and working abroad.